NAVAL POSTGRADUATE SCHOOL
MONTEREY, CALIFORNIA

THESIS

THE IMPACT OF CASE TOOLS, ADA, AND SOFTWARE REUSE ON A DOD SOFTWARE DEVELOPMENT PROJECT

by

Loren J. Dugan

March 1996

Principal Advisor: Nancy C. Roberts

Approved for public release; distribution is unlimited.

19960520 049 DTIC QUALITY INSPECTED 1

REPORT DOCUMENTATION PAGE

Form Approved OMB No. 0704-0188

Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instruction, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188) Washington DC 20503.

1. AGENCY USE ONLY *(Leave blank)*	2. REPORT DATE March 1996	3. REPORT TYPE AND DATES COVERED Master's Thesis

4. TITLE AND SUBTITLE **THE IMPACT OF CASE TOOLS, ADA, AND SOFTWARE REUSE ON A DOD SOFTWARE DEVELOPMENT PROJECT**

5. FUNDING NUMBERS

6. AUTHOR(S) Dugan, Loren J.

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Naval Postgraduate School
Monterey CA 93943-5000

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.

12a. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution is unlimited.

12b. DISTRIBUTION CODE

13. ABSTRACT *(maximum 200 words)*

Software continues to be the single greatest challenge facing DoD systems developers. The criticality of software as a integral component of system design, continues to grow in importance as DoD moves into the 21st Century. To meet this ever increasing challenge, software development organizations are incorporating new design requirements and practices into their development processes. These new requirements/practices, if properly implemented into the development process, can reduce software development and maintenance costs, and increase software quality and development productivity. This thesis investigates the impact of three management-selected software development requirements/ practices--CASE tools, Ada, and software reuse--on the progress of a particular DoD software development project, known as Project X. After a brief introduction, the thesis presents background literature on the three development requirements of Project X. The background literature is used to support the Project X case study. Information required for the case study is obtained through interviews with Project X affiliated development personnel. Interview results are analyzed and interpreted through a comparison with information found in the background literature. Results of the case study identify several problems with management's plan to implement the three development requirements into Project X. The thesis identifies specific root causes for the implementation problems, and makes recommendations to reduce the impact of these problems on Project X and other present and future DoD software development projects.

14. SUBJECT TERMS Software Reuse, AdA, Delta CASE Tool

15. NUMBER OF PAGES 134

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	UL

NSN 7540-01-280-5500

Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. 239-18 298-102

Approved for public release; distribution is unlimited.

THE IMPACT OF CASE TOOLS, ADA, AND SOFTWARE REUSE ON A DOD SOFTWARE DEVELOPMENT PROJECT

Loren J. Dugan
Captain, United States Marine Corps
B.S., University of Idaho, 1989
B.S., Southern Illinois University, 1989

Submitted in partial fulfillment
of the requirements for the degree of

MASTER OF SCIENCE IN MANAGEMENT

from the

NAVAL POSTGRADUATE SCHOOL
March 1996

Author: _____
Loren J. Dugan

Approved by: _____
Nancy C. Roberts, Principal Advisor

_____
Tarek K. Abdel-Hamid, Associate Advisor

_____
Reuben T. Harris, Chairman
Department of Systems Management

ABSTRACT

Software continues to be the single greatest challenge facing DoD systems developers. The criticality of software as a integral component of system design, continues to grow in importance as DoD moves into the 21st Century. To meet this ever increasing challenge, software development organizations are incorporating new design requirements and practices into their development processes. These new requirements/practices, if properly implemented into the development process, can reduce software development and maintenance costs, and increase software quality and development productivity. This thesis investigates the impact of three management-selected software development requirements/ practices--CASE tools, Ada, and software reuse--on the progress of a particular DoD software development project, known as Project X. After a brief introduction, the thesis presents background literature on the three development requirements of Project X. The background literature is used to support the Project X case study. Information required for the case study is obtained through interviews with Project X affiliated development personnel. Interview results are analyzed and interpreted through a comparison with information found in the background literature. Results of the case study identify several problems with management's plan to implement the three development requirements into Project X. The thesis identifies specific root causes for the implementation problems, and makes recommendations to reduce the impact of these problems on Project X and other present and future DoD software development projects.

TABLE OF CONTENTS

I. INTRODUCTION

This thesis investigates the impact of management-selected software development requirements/practices on the progress of a particular Department of Defense (DoD) software project. Managers are now having to implement new requirements/ practices to control spiraling costs and improve quality. The use of these requirements/practices are gaining acceptance in DoD, business, and industry.

A. GENERAL

The software revolution is upon us and it is increasing in momentum at an alarming rate. Today, practically all manufactured products that use electricity require some type of software program. These products range from simple toasters to large complex weapon systems. As the range, size, and diversity of these products increases, so to will the demand for quality affordable software. Software developers are finding it difficult to keep up with this increase in demand. Even the software products that are being produced are lacking in quality and maintainability. If the software development industry is to operate effectively in the face of this software revolution, then many organizations will have to make substantial improvements in their products and developmental processes.

Nowhere is software development more of a problem than in DoD. Like industry, the problem is not only in the software product itself, but also in how the software is developed. Problems in software development continuously result in low availability and reliability of software, cost and schedule overruns for projects, and the fielding of substandard systems. In a speech given in 1989, Air Force General Bernard Randolph, Chief of Air Force Systems Command, characterized software as the "Achilles" heel of weapons development. "On software schedules, we've got a perfect record: we haven't met one yet." Software development has been and

continues to be the single greatest problem area for program and project managers in the overall development of new systems. As the demand for quality software in DoD continues to rise, the problems associated with software development will only become more prominent. [Ref. 1]

Many of these associated problems can be related to the fact that software development is a dynamic and rapidly changing discipline. In spite of the speed with which software development is moving, many software professionals are still developing software using outdated, ad-hoc development processes. These processes lack structured development methodologies and are both time intensive and costly in terms of person-hours. Many organizations in the recent past have resisted change to a more structured and standardized environment for fear of losing their creativity and individuality. [Ref. 2]

As previously stated, changes in the software market and the industry as a whole, have forced many organizations to reevaluate their software development processes. The demand for complex integrated software applications is up 25% in some markets. However, the annual growth of software programmers is only 4%. [Ref. 1] This equates to a greater demand for software than the industry can provide. "If you project current trends in (both commercial and military) software supply and demand out to the year 2040, you find that every man, woman and child in the country will have to be a software programmer," says William Wulf, assistant director for computer and information sciences and engineering at the National Science Foundation. [Ref. 1] It is obvious that organizations must change to accommodate this surging trend.

To remain competitive, management has begun making changes in how their organizations develop software. Development processes are becoming more defined and better documented. The focus of process improvement is now on software

engineering principles rather than the ad-hoc practices of the past. Organizations are embracing advances in technology, and management remains committed to continuous improvement through accurate measurement programs. [Ref. 2]

DoD is also realizing the importance of improving the software development process. Defense budgets maybe shrinking, however the need for complex weapon systems and information management systems is still growing. Even more critical is the fact that hardware costs are going down and software costs are rising. As a result, DoD like the commercial industry, is also implementing technological changes and process improvements in order to achieve greater efficiency and productivity out of their software developers. Many of these process improvements are being implemented in the form of mandates. The Ada language was one such mandate. Other process improvements are implemented at the discretion of the individual software development managers with the support of centralized software development agencies. These include such practices as the use of Computer Aided Software Engineering (CASE) tools and software reuse programs. Whether mandated by DoD or merely supported through centralized agencies, a commitment to software engineering disciplines and process improvement are the answers to DoD's software development problems. [Ref. 3]

B. RESEARCH QUESTIONS

This research seeks to answer specific questions regarding the effect of selected development requirements on a particular software development project.

1. Primary Question

How did management's decision to implement -- the Delta CASE tool, the Ada programming language, and software reuse -- affect the development of Project X[1]?

---

[1]At the request of the developing organization, the true name of the project will not be given. Refer to Section D of this chapter for further information.

2. Subsidiary Questions

The following subsidiary questions are provided as support to the primary question:

- Why was the Delta CASE tool used and what was its impact on Project X in terms of productivity?
- Why was the Ada programming language chosen and what was its effect on Project X in terms of productivity, quality, and cost?
- What was the extent of software reuse in Project X and what was its effect on the project in terms of productivity and quality?
- What corrective actions, in reference to the three subject development practices, could have been implemented to improve the development process?

C. SCOPE OF THESIS

The main thrust of this thesis is to develop an understanding of the management requirements to use specified CASE tools, the Ada programming language, and software reuse, and how these requirements affected the development of a particular DoD software development project. This study will mostly emphasize the management issues of the project in terms of productivity and quality. Technical issues regarding the three development requirements, will only be discussed briefly, and only to extent that they provide clarity to the subject management issues.

The primary research question will be addressed through the subsidiary questions. Each of the first three subsidiary questions separately addresses the software development practices utilized in the subject project. The last subsidiary question addresses possible corrective actions in relation to the three development practices.

Industry, business, and DoD are all experiencing the same challenges and problems in developing low cost, quality software. They are also implementing similar development practices to correct these problems. As a result, background research in the three subject development practices will be comprised of DoD, industry, and business literary sources. This action will provide greater depth to the research.

D. LIMITATIONS

The software development project that is the subject of the case study is still in full-scale development. Certain issues in this project are considered sensitive both from a political and developmental standpoint. At the request of the project management, neither the project name nor the name of the developing organization will be given in this thesis. This action will provide the researcher more freedom to explore more of the controversial aspects associated with this project. More importantly, the anonymity of the project will allow management and workers to be more open and frank in their answers to interview questions.

E. BENEFITS OF THE STUDY

The full intent of this thesis study is to increase the awareness and knowledge of the researcher in the area of selected software development requirements/practices and their effect on software development projects. This study will provide additional benefit to organizations within DoD and industry who engage in software development programs. Software development is still the single greatest problem for project and program managers. The insights and lessons learned provided in this study could be used by project and program managers to better manage their software development programs.

F. STRUCTURE OF THE THESIS

This thesis is organized into the following six chapters:

Chapter I introduces the reader to the thesis subject area. Information in this chapter includes: general information, the primary and subsidiary research questions, the scope of the thesis research, the limitations of the thesis, the benefits of the study, and the structure of the thesis.

Chapter II provides the required background information on CASE tools, the Ada programming language, and software reuse. This information is used to support the case/project analysis.

Chapter III identifies the research methodology used in the case/project analysis.

Chapter IV provides personal background information on the interviewees and an overview of the case/project. The results of the interviews are provided. These results are provided as points of agreement and disagreement among the interviewees. All relevant interview results not provided in the points of agreement and disagreement section, are presented in a follow-on section.

Chapter V provides an analysis and interpretation of the results of the case study as these results pertain to literature presented in Chapter II.

Chapter VI provides answers to the subsidiary and primary research questions. This chapter also provides recommendations for further research and concluding remarks.

II. LITERATURE REVIEW

This chapter provides a literature review of three key development requirements utilized in the planning and management of the software development process. The development requirements presented in this chapter correspond to the three development requirements mandated for the thesis case study project. Each of these requirements had varying degrees of effect on the case study project in terms of software quality and development productivity. Information presented on these three development requirements is used in Chapter V to analyze and interpret the case study interview results. The information presented in Chapter V is used to answer the primary and subsidiary research questions of this thesis. Essential background information on each of these three development requirements is presented as separate sections within this chapter. In order to provide greater depth and breadth in the background review of these three development requirements, research information presented will include industry, business, and DoD applications.

A. THE MANDATE FOR CASE TOOL USE

To generate any resemblance of success in developing quality, reliable, and maintainable software, software development managers must remained focused in one key area, software engineering. A quote by Mosemann provides substantial reasoning as to why an engineering discipline is the key to successful acquisition and management of software intensive systems,

> ...we've got to adopt an engineering focus. We have got to concentrate on cost-effective solutions, solutions that are built from models, and on using capable, defined processes, rather than focusing on perfect systems that meet 100% of our wishes. Again, this is a management challenge, not a technical challenge. There's just no way to manage or

> to control the configuration, to control the side-effects, in these kinds of large software developments unless we use engineering discipline. [Ref. 3]

The push in software engineering activities is the key to success in software development and not merely a matter of inconvenience for software developers. Demand for quality, maintainable integrated software packages is growing faster than industry can produce. To remain competitive and capable of producing quality usable products, software development managers must look at current processes and practices, and find ways to improve them.

Advances in technology is at the center of software development improvement. According to Tate et al.,

> Technology is central to organized society's efforts to improve the lot of individuals and organizations, regardless of one's opinions of the success or failure of instances of technological application or of the ultimate nature of improvement. [Ref. 4]

Since the 1960s, software development processes and products have continuously improved as a result of the introduction of software engineering techniques in the form of new concepts, languages, methodologies, techniques, and tools [Ref. 4]. Computer-Aided Software Engineering (in the form of CASE tools) is one such area where technology and software engineering practices have combined to improve the development process and the quality of software products. Software development managers are now witnessing the inherent benefits of utilizing CASE tools, as they strive to implement software engineering practices in their development process.

1. CASE Tool Definition and Background

According to Fuggetta, "A CASE tool is a software component supporting a specific task in the software-production process." [Ref. 5] CASE tools are a subset of an overall technology known as CASE. CASE technology according to Sodhi, "encompasses a collection of automated tools and methods that assist software engineering in the phases of the software development life cycle." [Ref. 5] The other subclasses of CASE technology products utilized in the production-process realm of software development are Workbenches and Environments. Fugetta classifies these three subsets of CASE technology products in the following way:

> Tools support only specific tasks in the software process.
> Workbenches support only one or a few activities.
> Environments support (a large part of) the software process. [Ref. 5]

As is apparent from these classifications, CASE tools are used to form different types of Workbenches and Environments.

CASE tools, according to Fuggetta, can be separated into the following classes: editing, programming, verification and validation, configuration management, metrics and measurement, project management, and miscellaneous tools. Editing tools comprise applications such as traditional text editors (word processors) and graphical drawing/painting editors. Programming tools are used to support coding activities such as coding and debugging, code generation, and code restructuring. Verification and validation tools are designed to carry out quality assurance tasks thereby ensuring that the output product functions as specified by the customer. Configuration-management tools coordinate and control the generation of a system from it's composed parts and includes areas such as change control, item identification, and configuration building. Metrics and measurement tools are used to collect data on

programs to be used for statistical analysis. Project-management tools are used to support a multitude of management functions to include project cost estimation and project planning. Finally, miscellaneous tools are those tools that are difficult to classify and include applications such as spreadsheets and hypertext systems. [Ref. 5]

Recent advances in technology and changes in computer architectures has brought about changes and a wide range of variations in CASE tools. CASE tool applications range from the early mainframe-based systems to the present desktop and client-server applications. They range in complexity from simple drawing tools (accomplishes a single, specific task) to the recently developed integrated CASE packages which are designed to automate the full development process throughout the software development life cycle. [Ref. 6]

Of course tied to this varying degree of complexity are varying degrees of prices for the CASE tools. Usually, the more complex the tool (in terms of capabilities), the higher the price. There are a myriad of vendors that offer CASE tool products in all price ranges and varying degrees of capability. The price and complexity/capabilities of CASE tools (in addition to other factors) will be prime considerations for software development managers when selecting an appropriate CASE tool. [Ref. 6]

2. Management Reasons for Case Tool Usage

A program manager, when confronting a software development project, is concerned with three general areas: cost, schedule, and performance. From a generic standpoint, all three areas have equal importance in producing a quality product, although certain projects may place a greater weight on one or more areas. Requirements/design specifications establish goals and thresholds for system performance within a specified set of conditions. Program managers must always

plan and coordinate the development process within an allotted time schedule to ensure the final product is completed by the deadline. It seems however, that cost provides the greatest significance as a constraint for management in the software development process. According to Jones, "cost overruns are a significant contributor to the problem of *Friction with Users* and the problem of *Friction with Senior Management*." Cost overruns are also associated with missed schedules and canceled projects. [Ref. 7]

The introduction of quality, applicable CASE tools in a software development organization is a method of minimizing the risk and uncertainty associated with cost, schedule, and performance constraints. Says Tate et al., "Ultimately, the motivation for tool use is economic--for competitive advantage." [Ref. 4] To achieve competitive advantage, an organization must produce, distribute and support a wider range of higher quality products faster and more efficiently than their competitors. In terms of Government software development, the key is to achieve the greatest capability for the least cost. The objective of CASE tools is to improve the software development process through the use of software engineering principles and greater automation of "software production." [Ref. 4]

Industry, commercial business, and DoD are all experiencing similar problems in software development: rising costs of programmers, shortages of qualified software engineers, an increase in demand for large integrated software products, and development backlogs. Organizations are seeing the benefits of introducing CASE tool technology into the development process. DATAMATION'S Technology Implementation Index states that 30% of surveyed readers are currently using CASE tools and another 54% are either reviewing or in the process of introducing CASE tools. Says C. Graham, "CASE technology provides managers with the greatest potential for improving productivity because it employs a step-by-step methodology

for software development and maintenance...it automates some of that process, while also preserving a degree of flexibility." [Ref. 8]

Defining a process and improving the productivity, consistency, and repeatability of that process while preserving development flexibility are strong reasons for introducing CASE tool technology into a software development organization. With a properly implemented CASE tool, a software development activity could achieve greater productivity and software quality with fewer development personnel. These actions would serve to reduce the problems that organizations have experienced with software development.

3. Beneficiaries and Potential Users of CASE Tools

Increasing the efficiency and effectiveness of the development process are key advantages of introducing CASE tools into an organization. However, economics is the deciding factor when determining if an organization would benefit from the use of CASE tools. Will the payoff benefits associated with increased efficiency and effectiveness be greater than the investment cost of introducing a CASE tool(s) in an organization? Is an organization's culture commensurate with the usage of automated CASE tools? These are questions to be answered by the software development managers in an organization before a decision is made as to whether or not to buy and utilize CASE tools.

a. The Economic Paradigm

As previously stated, economics is the ultimate motivation for implementing CASE tools. In business and industry, organizations want to maintain a competitive edge over their competition. DoD organizations want to achieve the most for each allocated budget dollar (especially as allocated budget dollars continue to decrease). The focus in organizations is not just on production costs, but on the full life cycle cost of producing and maintaining software products. One old adage of

producing a product at the lowest price has been replaced with creating products that have greater capability, are flexible and adaptable to different environments, can be easily updated, have reduced development times, have high reliability, and can be easily maintained. All organizations developing software strive to achieve these attributional goals in their products.

It is the size of the organization, the quality of it's software engineers, and the size of the products that are built that determines whether CASE tools are utilized and seen as a cost effective approach to developing quality software products. The use of CASE tools is especially prevalent and necessary in the large commercial software producers such as Microsoft, Borland, and Lotus. Says C. Jones, "...hundreds of software personnel have learned the hard way that low-quality products do not sell well." [Ref. 7] Quality in software is viewed as important as product development time and usability of products in maintaining a competitive edge for these large organizations. Mid-sized commercial software producers, according to Jones, "are at serious risk due to lack of software quality automation." Most of these companies are in direct competition with the large commercial software producers. In most cases they lack the resources and capital to obtain quality CASE tools. Compounding the problem further is the difficulty mid-sized organizations have in staffing only quality software engineering personnel, especially when the personnel strength of the company exceeds 50 software employees. [Ref. 7]

The prevalence of CASE tool usage is determined not only by the size of the organization and quality of its personnel, but also by the size of the product being produced. When looking at all development projects, design CASE tools are utilized less than 50% of the time. More importantly, usage decreases to less than 30% for projects less than 500 function points (function points is a metric used to measure the size of a software product) and usage increases to more than 60% for

projects greater than 5,000 function points [Ref. 7]. Organizations with less than 25 software developers (with all considered quality software engineers) can many times produce quality products of less than 2000 function points without the use of CASE tools [Ref. 7]. However, adequate CASE tool usage would increase productivity and cost effectiveness in these organizations, especially if not all employees were considered quality experienced software developers. The reason that CASE tools are not found as much in the smaller organizations is a lack of investment capital similar to the problem found in mid-sized organizations. Small and mid-sized organizations as well as large companies must do a cost-effectiveness analysis to determine if the investment amount spent on purchasing and training personnel to use a selected CASE tool will be less than the money saved through increased productivity and product quality. As the number of CASE tool vendors increases along with a wider range of available products and prices, the decision to acquire CASE tools will be easier for these organizations.

b. The Cultural Paradigm

The culture of the organization is the other deciding factor as to whether CASE tools are utilized or should be utilized in an organization. Cultures that easily adapt to CASE tool usage are those that embrace change in the form of technological advances and improved development processes [Ref. 7]. They understand that competitive advantage can only be achieved through sound management principles, innovation, introspective analysis, and a common goal for all personnel within the organization. Employees work effectively in teams and view all jobs within the organization as essential in obtaining the goal of the organization.

Researchers have found that there are several organizational and cultural factors that relate to the degree of CASE tool usage in an organization. One such empirical investigation by A. Rai and G.S. Howard looked at the level of CASE

tool usage in information system departments (ISD) in relation to five factors: organizational environment, user characteristics, organizational processes, organizational structure, and task characteristics [Ref. 9]. Within the five factors, seven independent variables were examined in terms of their degree of effect on CASE tool usage. The independent variables were as follows: the perceived threat to ISD survival, the degree of in-house expertise in structured methodologies and software engineering, organizational size, the degree of CASE technical support available, the degree of CASE championship (a person in the organization who is a strong advocate and promoter of CASE tool usage), the degree of top management support, and the degree of job/role rotation in an ISD. The results of the study are as follows:

1. There is a positive relationship between the degree of perceived threat to an ISD's existence and degree of CASE usage for systems development.

2. There is a positive relationship between the degree of methodology expertise in the ISD and degree of CASE usage for systems development.

3. There is a positive relationship between the natural logarithm of size of an ISD and degree of CASE usage for systems development.

4. There is a positive relationship between the degree of CASE technical support available in an ISD and degree of CASE usage for systems development.

5. There is a positive relationship between the degree of CASE championship and degree of CASE usage for systems development.

6. There is a positive relationship between the degree of top management support for the information system (IS) function and degree of CASE usage for systems development.

7. There is a positive relationship between the degree of job/role rotation in an ISD and degree of CASE usage for systems development. [Ref. 9]

The researchers in this study urge caution when generalizing the findings in the study until an adequate replication of the study has taken place. In practice, these results can be used by software development managers as guidelines to organizational and cultural change. The movement of change is towards total acceptance of innovation and CASE tool usage [Ref. 9] It is also important to note that until organizations strictly follow one system development methodology, the introduction of CASE tools will not benefit the development process. The adherence to one development approach will allow greater acceptance of CASE tools by software developers. [Ref. 10]

B. ADA AS A LANGUAGE OF CHOICE

Since the development of the computer, we have witnessed a remarkable evolution in the writing of software applications. The evolution of computer programming languages is one area of software development that has witnessed some of the greatest changes. The first and most basic of all languages used to write software applications was machine language. Machine language consists of commands and/or data written in binary numbers. Binary numbers (machine language), consisting of different combinations of 0's (lower voltage) and 1's (higher voltage), is the only language that can communicate directly with the computer. Every type of computer interprets/reads these combinations of 0's and 1's differently in carrying out specified instructions or reading data. [Ref. 3]

Assembly language was the next or second generation of language to follow machine language. Unlike machine language, assembly language utilizes a shorthand

notation of letters and numbers which are easier to use. Each shorthand notation command corresponds to a binary number that the machine can understand. [Ref. 3]

Third generation languages, better known as higher order languages (HOLs), followed assembly language in the evolutionary development of programming languages. HOLs were significantly easier to write and read than assembly language because of their close resemblance to spoken language. Also fewer statements were required for any given function which increased programmer productivity [Ref. 3]. Fourth and fifth generation languages have been developed, however, their use is beyond the scope of this study.

The sheer number of first, second, and third generation languages available and in use since the 1970s is staggering. The single greatest concern with having so many different languages lies in the area of standardization. In no area is the lack of standardization greater then in HOLs. "When HOLs first became popular, hundreds of languages were developed, many for specialized applications on specific computers," says the Air Force Software Technology Support Center (STSC) [Ref. 3]. This was especially prevalent in DoD programming during the 1970s where there were more than 450 languages in use. There was no control or standardization which resulted in dozens of dialects forming from these base languages [Ref. 11]. To directly resolve this program language standardization problem, DoD adopted MIL-STD-1815A, the Ada programming language [Ref. 3].

1. Ada Background and Policy

Ada use was mandated by DoD in 1980 to standardize software development. According to the Air Force STSC, the use of Ada as a standardized language provides three benefits:

> First, software personnel in DoD and within its contracting community had become fragmented with the proliferation of languages used for

> military software prior to 1973. This meant software professionals were very specialized and could not move readily from one project to another. Second, so many languages meant that DoD had great difficulty transporting software across computer environments. In addition to the cost of rehosting software, the diversity of development languages meant DoD could not easily employ the software it had as a capital stock of predeveloped, pretested software components available for reuse in other systems. Finally, the large number of languages meant that few commercial software tools were available for any given language. With the consolidation of the considerably large DoD software market into one language, tool makers would have a larger, single market upon which to concentrate their efforts. They would, thus, have the incentive to produce more and better tools competitively priced for DoD. [Ref. 3]

These three benefits were strong qualifiers for the Ada mandate. Additionally, the Ada language mandate was designed to advance key software engineering principles such as reusability, portability, maintainability, and reliability. In 1983, following the mandate, the American National Standards Institute approved the Ada language thereby ensuring Ada's credibility as a certified programming language. In 1987, Ada was approved by the International Standards Organization. [Ref. 12]

The adoption of MIL-STD-1815A was just the beginning of a progression of Ada policies implemented by DoD. In 1987, DoDD 3405.1 (Computer Programming Language Policy) and DoDD 3405.2 (Uses of Ada in Weapon Systems), were established. [Ref. 3] According to the GAO, "these defense directives declared Ada the single, common computer language for use in its automated weapon systems, and information systems except where another language could be demonstrated to be more cost effective." [Ref. 12] Under their interpretation of these directives, the Air Force and Army ordered the use of Ada for both weapon systems and information systems. In contrast, the Navy required Ada for weapon systems, but allowed any approved HOL for its information systems. [Ref. 12] To remedy this interpretation

problem, President Bush signed Public Law 101-511 in November 1990, requiring all DoD software be written in Ada. In June 1991, DoDI 5000.2 was established and stated that Ada is "the only programming language to be used in new defense systems and major upgrades of existing systems." [Ref. 3]

For the Air Force, Colonel Richard R. Gross explained the importance of this policy.

> If I had to paraphrase the policy in a couple of sentences, these are the ones I'd use: "We think Ada is smart. Therefore, use it."...It's not an "Ada for Ada's sake" policy: We in the Air Force view Ada as a means to an end, and not an end in itself. [Ref. 3]

In June 1995, twelve years after the original Ada language was standardized, the updated and improved Ada 9X version was approved. Ada 95, as it is now called, has become Federal Processing Standard 191-1. The newest Ada version was approved by the American National Standards Institute in 1994 and the International Standards Organization in early 1995.

2. Areas and Domains Best Suited for Ada

Ada is becoming the language of choice for weapon systems designers and information systems builders. Ada is the most widely used programming language in weapon systems designs and is second to COBOL in other DoD development domains [Ref. 11]. The DoD mandate was directly responsible for Ada's increased usage among DoD organizations and defense contractors. Ada usage in Real-time weapon systems, C^3 systems, and information systems continues to grow [Ref. 3].

DoD organizations and defense contractors are not the only driving force behind Ada's increase in popularity. System developers in the commercial sector have increased their use of Ada by an astonishing rate. The U.S. market in 1989 was

$975 million. It was projected that the market in 1995 would be somewhere between $2.4 and $9 billion. "This growth in Ada use has occurred because major corporations are increasingly embracing Ada where safety and reliability are their bread and butter." [Ref. 3] Safety and reliability are also two of the most important reasons why DoD system developers embrace Ada. Lt. General Ludwig states:

> Ada is the language of choice where human life is at stake. Sometimes software people spend so much time staring at their computer screens, they forget that our fellow warriors are strapping on your software and putting their lives on the line. When human life is at stake, the question is not what language is the easiest to use or the most popular, it is what language will give us the highest safety and reliability. NASA, the FAA, commercial airliner companies, and many others all chose Ada for the same reason. [Ref. 3]

The versatility and emphasis on safety are not the only benefits or attributes of Ada. Ada is much more than merely a standardized language for DoD applications. According to the SEI:

> Ada is unlike other languages, however, in the degree to which it fosters and supports the practice of software engineering principles. These design principles are believed to lower software development costs, increase software quality, and lower maintenance costs, especially for large or complex systems. In effect, these features and the structure of the language make it easier to develop software that is more understandable and more maintainable. [Ref. 3]

Lower maintenance costs are one of the primary reasons why DoD will continue to push the Ada mandate for all developmental projects. The advantage in Ada's usage within large or complex systems will become readily apparent as older systems, both DoD and commercial, are updated, expanded, and/or combined. These advantages

are readily apparent in Ada 95. The multi-language interface and object oriented capabilities of Ada 95 enable the integration of several different products/systems. [Ref. 11]

These attributes and many more are reasons why Ada is the language of choice for real-time weapon system development. Ada is considered to be in a class of "real-time languages" which makes it especially important for the development of critical communications and military projects. The inherent characteristics of Ada as a real-time language make it more unique than other general purpose-languages. These characteristics, according to Hinden et al., include: "multitasking capability, constructs to directly implement real-time functions, and modern programming features that help ensure program correctness." Multitasking is important because it allows the system to respond simultaneously to asynchronous events. [Ref. 13]

The inherent advantages of Ada as a real-time system integrator are readily apparent in the F-15E Strike Eagle. The aircraft is referred to as "Ada's Eagle by the joint program team who built the Ada-based Integrated Control System (ABICS) for her athletic airframe." Consisting of a whoping 2.3 million lines of code, the ABICS easily allows the "back-seater" to select targets, plot intercepts, and choose munitions. The Ada based software also controls flight-critical systems in the aircraft to include: navigation, flight controls, and sensor systems. As Lt. General Ludwig remarked, "It is truly amazing that two people in that F-15E can accomplish in one sortie what it took a thousand men in 100 bombers, with many lost lives on every mission, to do in World War II." [Ref. 3]

As technology has progressed in software and hardware, so too has the similarities within the different software development domains (e.g., weapon systems and MIS). When Ada compilers became available, commercial and DoD MIS developers were able to take advantage of Ada's multi-tasking and system integration

attributes. As a result, large Ada based MIS projects have become abundant throughout DoD and the commercial sector. The Marine Corps paycheck upgrade systems, and the Army's Standard Finance System Redesign are examples of DoD applications. Ada based MIS applications in the commercial sector include: the Multi-state payroll systems, and the On-line banking systems for the Nokia Information Systems. Ada's ability in the MIS domain will only get better since the newest upgrade, Ada 95, contains special features that allows for even more capability in MIS applications. [Ref. 3]

The bottom line is that software developers in all domains can achieve success with Ada. Organizations that have migrated to Ada and implemented dependable software engineering techniques have seen increased programmer productivity, reduced defects, increased performance, achieved more accurate life cycle estimates, and reused more existing code [Ref. 3]. Table 2-1 represents the results of a data collection effort on 75 industry Ada programs.

3. Marketing and Consensus Building for Ada Usage

The standardization of the Ada language required an incredible consensus among software development professionals within 21 voting countries [Ref. 14] The fact that the representatives from these countries were able to systematically approve Ada, speaks a lot for the capability and versatility of the language. However, consensus and approval from the professional organizations will not alone guarantee the success of Ada. Software development organizations must choose to use Ada because it is truly the best choice among programming languages. To establish this support, software professionals, software development organizations, and educational institutions must embrace Ada as a viable programming language and not merely a DoD pushed mandate.

Table 2-1. Industry Experience with Ada [Ref. 3]

PRODUCTIVITY	- 10-20% degradation during first project - 20% improvement attainable on subsequent projects versus other languages - 17% improvement attainable on Ada projects through environment/ tools integrated with object-oriented methods
QUALITY	- 20-30% fewer defects than other languages on first projects - Maintainability significantly improved with 10-20% lower cost
PERFORMANCE	- Ada programs 10% smaller than other languages using conventional packaging methods - Benchmarked Ada programs ran as quickly as others in 70% of the cases
LIFE CYCLE TIME	- Ada had no effects on life cycle time - Distribution changed from 40% design, 20% code, 40% test (traditional) to 50% design, 10% code, 35% test (with Ada) - Incremental or spiral models reduce the life cycle 20-30%
REUSE	- 18% average today compared with 10% 3 years ago - (10% needed to break even given cost of reuse)

The largely artificial beginnings of Ada were in many ways causes for its lack of support and use among software professionals. DoD's mandate for the development and subsequent push of Ada was executed separately from the requirements and demands of the commercial software industry. The push for Ada was seen as a threat to the programmer's individuality and creativity in both DoD and industry. The attitude of some software engineers was that C++ was the only "serious programming language." C++ has generated a large following in the commercial sector which makes it increasingly difficult for Ada to break in. [Ref.15]

Ada did not have the commercial industry marketing forces supporting its use. Ada tools and compiler vendors failed to advertise their products which resulted in less visibility, less market share and ultimately less availability for Ada products. Additionally this lack of support has resulted in a shortage of qualified Ada software engineers. In the past, the small number of educational institutions which offered an Ada programming curriculum could not produce enough qualified engineers to fulfill the requirements of DoD and industry. DoD has realized that the only way to correct these shortcomings and make Ada the programming language of choice among DoD and commercial sector organizations is through an aggressive marketing campaign. The remainder of this section will briefly discuss the separate areas targeted for the marketing campaign to include strategy and reasoning. [Ref. 16]

The goal of making Ada 95 a multi-standardized programming language was an investment for the future. In order to generate a large support network for the language, the international software community had to be intimately involved throughout the development process. This deep involvement made it very difficult to achieve a consensus on proposed changes. However, when consensus was achieved (through compromise), then everyone had buy-in or support for the final product. [Ref. 17]

The results of the Ada 95 international project are notable. Says S.P. McCarthy:

> Ada supporters hail the new version of the programming language as robust and worth a second look by developers. Its built-in help levels are higher than C or C++, its competitors. Ada 95's specific features were developed in accordance with 750 user recommendations. Support for core business capabilities, rather than military applications, is also ensured by the new system. Initial versions of Ada 95 compilers show strong support for a variety of Unix and PC platforms. [Ref. 15]

Feature updates not withstanding, the fact that Ada 95 has achieved approval as a standard language from the National Institute of Standards and Technology, the American National Standards Institute, and the International Standards Organization can help generate the respect of the user community and promote the use of Ada in future projects.

DoD with the help of the Department of Energy (DOE) has recently devised a program to "incentivize" companies to develop various commercial Ada products. The program called Ada Technology Insertions Program-Partnership, provides a $2 million award to eight to 10 organizations. All Government awards must be matched by the organization and will be used to fund Ada tool development projects. Sponsors of the program, Idaho National Engineering Laboratory (INEL) and the Ada Joint Program Office (AJPO), have envisioned two primary goals for the program: create more tools for the Ada language while at the same time expanding the number of users. "The result of the partnerships will be enhanced tools which will make the utilization of Ada easier and, by definition, the user base is going to expand," said Doug Colonel, a program manager with Lockheed Idaho Technologies Co. [Ref. 18]

The INEL and the AJPO are also sponsoring a program to increase the pool of available Ada programmers. As witnessed in the past, qualified Ada programmers were difficult to find. This program proposes to correct this deficiency by providing 14 grants to U.S. universities for the sole purpose of educating future Ada programmers. [Ref. 18] The grants, totaling $800,000, will be used to develop Ada curriculums with the greatest significance on Ada 95, computer science, information engineering, and other engineering related classes. [Ref. 19] Sponsors hope these efforts will not only increase the availability of qualified programmers, but will also generate greater acceptance of the Ada language within the academic community.

Research has shown that Ada is generating greater interest among users and software development organizations. The aggressive marketing programs established by the INEL/AJPO were undertaken to develop the commercial support tools and personnel resources required to meet this new found interest. Will these programs and Ada's renewed interest be enough to generate increased supply and demand in the commercial sector? Only the future can provide the answer to that question.

C. SOFTWARE REUSE AS A TIME AND COST SAVER

As competition becomes greater in the software development industry, management must explore new ways of creating cost and time savings in their development methods. The common theme in the industry is to produce a better product within a shorter period of time. As with any manufacturing or development organization, the way to accomplish these savings is through process improvement. The utilization of software reuse is a process that has proven to reduce development costs and time in software builds.

1. Origins of Software Reuse

Software reuse, said R. Prieto Diaz, "is the use of existing software components to construct new systems." Software components in this definition

include much more than just pieces of source-code. It pertains to all other software components that are generated during the intermediate level of development to include: requirements, specifications, designs, and architectures. All of these components have the potential for reuse in future development projects. [Ref. 20]

Software reuse has a rather long history. The first incident of software reuse took place back in 1944 when a DoD programmer wrote a short routine to compute the sine of an angle, something useful for plotting the trajectories of missiles [Ref. 21]. Approximately 24 years later, software Reuse was introduced as a formalized concept by Dough McIlroy. In his paper titled "Mass-Produced Software Components," McIlroy proposed "an industry of off-the-shelf, standard source-code components and envisioned the construction of complex systems from the small building blocks available through catalogs." McIlroy's idea became the vision of Systems Development Corp. who registered the term "software factory" as a trademark in 1974. [Ref. 20]

The mid to late 1970's brought even more interest in reuse. Robert Lanergan of the Raytheon Missile Division, started what was considered to be the first formal organizational reuse project. Reuse was also beginning to generate interest in the academic community. Several research papers were written on reuse with emphasis being placed on both technical and nontechnical issues. [Ref. 20]

Reuse made tremendous leaps during the 1980s with the development of large-scale reuse programs. Several governments and industry demonstrated a great deal of commitment to reuse as the inherent benefits became more widely known. Advancements in reuse were made in areas such as libraries, classification techniques, the creation and distribution of reusable components, reuse support environments, and reuse in corporations. In 1988, Vic Basili broadened the definition of reuse to include the "use of everything associated with a software project, including knowledge." The new broader definition generated research in other nontechnical areas. Today, the

emphasis is being placed on the integration of both technical and nontechnical factors into a defining concept called institutionalized reuse. [Ref. 20]

2. Categories of Reuse

In order to promote or implement reuse in an organization, management must have a general understanding of the different categories or types of reuse. Each category or type of reuse provides a different perspective for management on how reuse should be viewed or utilized in an organization. Some organizations may choose to implement certain categories of reuse, where as others may desire to implement all categories. The beauty of reuse is the inherent flexibility it provides to management in saving time and money.

In this section, we will identify one industry method currently used to identify the different categories of reuse. This method was introduced by Ruben Prieto-Diaz in his article titled "Status Report: Software Reusability." In his article, he divided reuse into six "conceptually independent facets," (see Table 2-2). Each facet contained two or more methods or categories of reuse. Prieto-Diaz identifies each facet as follows: "The by-substance facet defines the essence of the items to be reused; by-scope defines the form and extent of reuse; by-mode defines how reuse is

Table 2-2. Facets of Reuse

By-Substance	By-Scope	By-mode	By-technique	By-intention	By-product
Ideas, concepts	Vertical	Planned, systematic	Compositional	Black-box, as-is	Source code
Artifacts, components	Horizontal	Ad-hoc, opportunistic	Generative	White-box, modified	Design
Procedures, skills					Specifications
					Objects
					Text
					Architectures

conducted; by-technique defines what approach is used to implement reuse; by-intention defines how elements will be reused; and by-product defines what work products are reused." [Ref. 20]

Each type of reuse found under the six different facets has its own unique characteristics. Idea reuse looks at applying formal concepts as a solution to a particular class of problems. Artifact reuse is a focus back to the past where "software factories" develop new systems from a collection of software components. Procedures reuse concentrates on "formalizing and encapsulating software-development procedures." Vertical reuse concentrates efforts in one domain or application. Horizontal reuse specifies the use of standard components across domains or applications. Planned reuse is a deliberate, formalized plan for the systematic use of reuse in an organization. Ad-hoc reuse is considered "opportunistic" reuse where individuals select components from a central repository for use in a new project. Generative reuse identifies reuse "at the specification level by means of application or code generators." Black-box reuse identifies the reuse of unmodified components. White-box reuse denotes the modification or adaptation of components for reuse. [Ref. 20]

The types of products found within the by-products facet is only a partial list of products that are used for reuse. Each product has its own individual characteristics and benefits. Source code reuse is the simplest form of reuse. As reuse tools evolve, there will be less emphasis on code reuse. Design reuse is a new technique that promises a "higher payoff" than code reuse. Specifications reuse has the potential for the "highest payoff" and is associated with generative reuse. Object reuse, in combination with object oriented methods, is seen as the technique of the future. Text reuse looks at "integrating reusable text" with other products in the development project. Architect reuse is the largest of the listed products. In architect

reuse, "application domains are analyzed to find generic designs that are then used as basic templates for integrating reusable parts or for developing specialized code generators." [Ref. 20]

C. Jones identifies five other reuse products: data, estimates, human interfaces, plans, and requirements. Data reuse is a form of artifact or component reuse and is similar to code reuse. Estimate reuse signifies using previous projects to estimate the cost of a similar future project. Human interface reuse attempts to increase familiarity with new systems through the standardization of interfaces such as installation procedures and function keys. Plans reuse is a method to utilize plans from previous projects for use in future project planning. Lastly, requirements reuse identifies the reuse of user requirements from project to project. [Ref. 7]

3. Potential Benefits of Reuse

Software development organizations implement reuse programs for one simple reason, competitive advantage. The number of companies developing software is increasing considerably. Many of these companies are competing within the same system domain. [Ref. 22] To remain competitive and possibly increase their share of the market, many organizations implement formal software reuse programs. Reuse, when implemented as a formal systematic program, has the potential of increasing productivity, increasing quality, reducing cost, and reducing development time.

The key point to consider in the last statement is that reuse has the potential of creating all those benefits. Says D. Kaspersen, "The impetus is clearly economic -- designing and implementing software for reuse is a substantial initial investment that has the potential to reduce development costs and schedules significantly." In other words, will the reduced costs of utilizing reuse in the development of new systems be greater than the capital investment of implementing that reuse program. [Ref. 23]

Organizations that continuously develop new software applications have the potential of benefiting from reuse if they are committed to developing reusable software components/products/materials. The catch here is that reusable components tend to cost substantially more to build. According to C. Jones, "the cost of developing and certifying a reusable module of 5 function points is about $5,000" compared to $2,500 for an average module of 5 function point. Reusable modules must be certified to a status close to zero defects which accounts for the increase in development cost. A $5,000 reusable module would save development costs if it was reused two or more times. [Ref. 7]

Quality becomes a benefit of reuse since the number of defects found in reusable modules is substantially less than the average module. The use of reusable modules along with specialized modules, will result in higher quality for the project. In an internal study done by Hewlett-Packard, researchers found the defect rate for reused code to be about .9 defects per thousand noncomment source statements (KNCSS). This is compared to 4.1 defects per KNCSS for new code. In a software product containing 68% reused material, they found 2 defects per KNCSS. The result was a 51% reduction in the number of defects when compared to new code. [Ref. 24]

Higher productivity in software development is another advantage of reuse. Cost savings and shorter development schedules are by-products of increased productivity. Higher productivity is much more apparent in reuse rather than constructing customized components. Developing customized components for each project can be very time consuming. Besides the actual designing and coding activities, there are the requirements for testing and debugging. The testing and debugging iteration can take place several times before a component is ready for use. Through the use of reusable components, programmers will be able to develop a

system much faster since the required components have already been developed, tested and debugged. According to C. Jones, "applications where reusable designs, code, documents, and test material exceeds 50% of the total volume often have net productivity rates in excess of 35 function points per month." This is significantly greater than the U.S. average productivity rate of 5 function points per month. [Ref. 7]

Another study done by QSM Associates, Inc., supports the findings of C. Jones. In this study, the results of a sample of 15 software development projects employing reuse were compared with the results of 3800 projects not employing reuse. The results of the study were rather remarkable. An average size project for both groups was 133,000 lines of code. The reuse projects took approximately 5.3 months to build versus 18 months for the nonreuse group. This equates to a 70% reduction in development cycle time. Furthermore, the reuse projects required 27.9 person-months versus 179.5 person-months for the nonreuse projects, an 84% reduction in cost. [Ref. 25] The cost savings and increase in productivity found in this study provide strong support for the implementation of reuse in a software development organization.

4. Implementing a Software Reuse Program

The advantages and benefits of software reuse are readily apparent. Developing a formalized reuse program and remaining committed to that program is the best way an organization can achieve success through software reuse. To successfully implement this program, there are certain issues or barriers to software reusability that must be overcome. This section will discuss those issues/barriers, and then provide a plan for the development of a quality reuse program.

The first barrier regards quality. Simply stated, for reusable materials to be economical beneficial, they must be of the highest quality. A reusable module that

approaches zero defects will be easier to implement in a project than a module with a greater number of defects. An organization must be committed to quality for reuse to be cost effective. [Ref. 7]

Economics, in the form of a cost-benefit analysis, is the second barrier to overcome. The cost of constructing reusable modules is substantially more than standard modules. A cost-benefit analysis will show whether the reduced costs of reusing modules will be greater than the increased costs of producing them. [Ref. 22]

Management support is the third barrier that must be overcome. Without substantial support from the top-down, a reuse program is dead on arrival. Management must be willing to forego returns on the investment of reuse. In addition to the increase in cost per reusable component, other investment costs include, educating personnel, "creating and managing incentives," and the possible hiring of reuse support personnel. Management must also bring about change in the culture of the organization to embrace reuse. These actions will require a long-term commitment from management in order for the program to be successful. [Ref. 22]

The fourth barrier to software reuse is the lack of effective planning. Management must plan for reuse well in advance by ensuring that adequate materials and procedures are in place. These materials and procedures include: planning and estimating tools, a library of reusable materials, company standards and procedures, and design methods. [Ref. 7]

Finally, for a reuse program to be effective, some type of measurement system must be used. There must be a way to measure the benefits of reuse in reference to the amount of reusable materials used in development. Three effective measures were explained in the preceding sections, one each for cost, quality, and productivity. Each of these measures provides an indication of the effectiveness of the reuse program and provides management with useful information for making critical decisions. [Ref. 22]

Once these barriers to software reuse have been addressed and properly resolved, then the implementation of a quality reuse plan will be easier. There are an infinite number of ways to formulate a reuse plan. A working group at Motorola devised the following plan to facilitate a software reuse program.

1. Find a champion for each reuse role in the organization
2. Provide personnel with the required training
3. Analyze your product target domain
4. Have the reuse group establish department standards
5. Obtain state-of-the-art tools
6. Develop expertise on these tools
7. Populate a reuse library or repository
8. Obtain the initial baseline data
9. Train other personnel to use the tools
10. Start using reusable materials in new projects from the first day
11. Ensure there are adequate incentives for reuse [Ref. 26]

This list is not all inclusive. However, it does provide a solid baseline from where to start formulating a plan. One important point to remember is that no program will be ultimately successful without the full support of management.

5. Reuse in DoD

The military is one area where software reuse is embraced whole-heartedly. This devotion is evident in DoD's continued commitment to Ada 95 and in Integrated

CASE. [Ref. 11] Through the use of a standard language, a sound development methodology and standardized tools, reuse within DoD will become more prevalent.

In addition to their commitment to Ada and CASE technology, DoD has implemented several programs to aid the development of reuse programs in the various military services. One such program, Software Technology for Adaptable, Reliable Systems (STARS), provides transitional support to DoD organizations incorporating a software reuse program. Another program, Asset Source for Software Engineering Technology (ASSET), provides a "nationally distributed network of reuse libraries," and serves as a "central exchange for reusable software assets." [Ref. 3] These programs and many others serve to focus efforts in one area, to explore and expand the use of software reuse in DoD. [Ref. 7]

D. SUMMARY

This chapter has provided an overview of three key software development practices currently used in DoD, business, and industry. CASE tool usage, as a development practice, was defined and explained. A growing range of CASE tools are providing software developers with greater capability in the face of rising costs, shortages of qualified software engineers, and an increase in demand for quality software. There was also sufficient information presented on the indicators for potential CASE usage.

Ada, the DoD mandated language, was the second development practice introduced. Information presented focused on the reasoning behind the Ada mandate to include background and policy, and the issue of language standardization. Sufficient detail on the software domains best suited for Ada's programming characteristics was provided in the chapter. The details and importance of DoD's marketing strategy for increasing the user base of Ada was also presented.

Software reuse was the final development practice introduced in the chapter. The origins of software reuse date back to 1944. However reuse did not generate much interest until the 1970s. Today, software reuse is one of the best ways to reduce cost. This chapter presented the different categories of reuse and the potential benefits associated with each. There was also information presented on barriers to reuse and how these barriers can be overcome before implementing a formal reuse program. Finally information on software reuse in DoD was provided.

III. METHODOLOGY

The methodology employed in this thesis study consists of a case study. Research emphasis focuses on qualitative rather than quantitative information. With the exception of a single case study project document, all information for this case study was obtained through personal interviews. Throughout the remainder of this thesis study, the case study project will be referred to as Project X, and the organization responsible for developing Project X will be referred to as Alpha Organization. This chapter provides necessary background information on the methodology used in this thesis study. The first section of this chapter provides information pertaining to the structure of the interview process to include: a brief description of the interviewees, a brief description of the interview questions, and the conduct of the interviews. Chapter IV provides in-depth personal information on each of the interviewees, and comprehensive background information on Alpha Organization and Project X. The second section of this chapter provides limitations of this interview process.

A. REASONING FOR THE INTERVIEW PROCESS

Most information required for this case study analysis has been obtained through personal interviews. Additional information, in the form of a single Project X document, was acquired from the Project Manager. Because of the extreme limitations of available Project X archival information, personal interviews were the only way to obtain the necessary qualitative information for use in the interpretation and analysis portions of this study.

1. Brief Description of Interviewees

Four persons in different hierarchial levels of Project X and Alpha Organization are interviewed. During the time of this case study analysis, these four

interviewees were the only personnel available who had knowledge about the three development requirements of Project X. The persons interviewed (identified by their billet titles) are as follows: Programmer One and Programmer Two, the Project Manager, and the organization Software Coordinator. Programmer One, Programmer Two, and the Project Manager all were directly involved in the development of Project X. The Software Coordinator was mostly responsible for formulating software development strategy and incorporating change into the organization's software development process. Each of the four interviewees provides a different point of view on the three development requirements and their affect on Project X.

2. Interview Questions

All questions asked in the interviews came from a list of 51 questions found in the appendix. Thirty-seven questions from the list were given to all four interviewees. Five questions were given specifically to both programmers. The last 19 questions were given specifically to the Software Coordinator and the Project Manager.

The 37 questions common to all interviewees requested individual background information such as education and work experience in general software development. The emphasis was to gather as much historical information as possible in this area to establish a level of credibility for each interviewee on the use of CASE tools, the Ada programming language, and software reuse. Also, within these 37 questions were questions asked regarding the perceived level of impact these three development requirements had on the project. The researcher was specifically looking for the following: the ease with which these three requirements were implemented, their applicability to the project, their contributions to the project, the level of management support given on these development requirements, and any associated lessons learned.

Five questions aimed specifically at Programmer One and Programmer Two looked at the effects of the three development requirements from a user/developer standpoint. The researcher was looking for the effect these requirements had on the programmer's ability to develop Project X software. Programmer opinions were also requested on areas such as: problems with the development requirements, more efficient substitutes to the chosen requirements, and the interaction effect of the development requirements on Project X.

The last 19 questions were directed only at the Project Manager and the Software Coordinator. These questions dealt with issues such as: the reasons for establishing these requirements, the problems in acquiring materiel for these requirements, the management plan for implementing these requirements into the software development process, the level or degree of management support given to these development requirements, the known or perceived problems with these development requirements, the development requirements' impact on the development of Project X, the interactionary effect of the development requirements, and the cost efficiency of the development requirements. The focus of these questions was on management issues relating to the acquisition and assimilation of the development requirements into the development culture of Project X and Alpha Organization.

All 51 questions relate to the literature background information found in Chapter II. The results of these 51 questions is provided in Chapter IV, and is separated between the responses of Programmer One, Programmer Two, the Project Manager, and the Software Coordinator. A separate section in Chapter IV provides points of agreement and points of disagreement among the four interviewees. These results provide the necessary information to support the analysis and interpretation

portions of this thesis study. Information from the analysis section is later used to answer the primary and subsidiary research questions.

3. Conduct of the Interview

The interviews were conducted separately face-to-face in the work spaces of each individual interviewee. Personal interviews were chosen as the case study research medium since there were only a small number of interviews required. The use of personal interviews allowed for a richer atmosphere for questioning and enabled the interviewer to observe any non-verbal responses to questions such as body language and gestures, etc. All questions for both the programmers and the managers were asked in the same order. A tape recorder was used to record the responses to the questions. This allowed the interviewees to answer the questions at their own pace and with less distraction from the interviewer.

B. LIMITATIONS OF THIS METHODOLOGY

There is an inherent limitation and a possible danger associated with the methodology of this case study. This limitation directly involves the method by which the interviewee sample was chosen. Since the four interviewees were not selected at random from all personnel associated with Project X, there is a chance the results could be biased against the true outcome of the project. Only through random selection of a population sample can a researcher limit the affect of extraneous variables on the outcome of the interview results.

In this case study, the researcher did not have the luxury of randomly choosing a representative sample of interviewees. The Software Coordinator was the only person in the organization responsible for developing the policy and implementation plans associated with the three development requirements. Her viewpoint was required to establish the level of support given to the three development requirements and Project X. The Project Manager was the only mid-level manager associated with

Project X. He was the only person available that could provide information concerning the overall effect of the three development requirements on Project X. Finally, Programmer One and Programmer Two were the only programmers available who had experience with the three development requirements, and were either still working or had previously worked on Project X. All other programmers were no longer part of Alpha Organization, and could not be obtained for interview. To best deal with the chance of bias, the researcher looked for common trends or a consensus in interviewee responses. The assumption is that a consensus increases the likelihood that the researcher has received quality information concerning the negative and positive aspects of the development requirements.

C. SUMMARY

A case study is the research methodology chosen for this thesis study. Most of the required research information on the three specified development requirements of Project X was obtained through four personal interviews with Project X affiliated personnel. Personnel selected for interview consisted of the Software Coordinator, the Project Manager, Programmer One, and Programmer Two. There was a total of 51 questions for all four interviews; 37 were given to all interviewees, 5 were given only to the two programmers, and the last 19 were given only to the Software Coordinator and the Project Manager. The personal interviews were conducted face-to-face with a tape recorder being used to record the responses. Using a tape recorder to record the responses provides for a richer and less distracting environment for the interview process. There is an inherent limitation with this case study since the personnel sample chosen for interview was not randomly selected. However, this is a limitation the researcher can not overcome. The four personnel interviewed are the only persons available with knowledge on the three development requirements of

Project X. Because of the potential for biased information, the researcher looked more for trends and consensus in the interviewee responses.

IV. INTERVIEW RESULTS

This chapter provides the results obtained from interviews with four personnel involved in the case study project. Essential background information on Project X will be given in Section A prior to introducing the interview results. The background section on Project X provides both a historical perspective and information on three specific development requirements. The development requirements of Project X are CASE tool use, the Ada language, and software reuse. These requirements are the focus of the interview results and this thesis study.

Interview results will follow the section on background information. These results will be listed in Sections B, C, D, and E. Section B provides background information on the interviewees. Section B also identifies points of difference and points of agreement between the interviewees on interview results relating to the Project X development requirements. Section C provides interview results that pertain to the impact or effect of the Project X CASE tool on the development of Project X. Section D provides interview results that pertain to the impact of the Ada programming language in the development of Project X. Finally, Section E includes all interview results relating to the effect of software reuse on the development of Project X.

Interview results on the development requirements presented in Sections C, D, and E are separated between the responses of the four interviewees: Programmer One, Programmer Two, the Project Manager (PM), and the Software Coordinator who manages all software development operations and computer related activities within Alpha Organization. The interview results are organized and presented in a way that facilitates analysis of information thereby allowing greater ease in answering the primary and subsidiary research questions. The commitment to anonymity requires

that the names of the interviewees, the project, and the organization developing the project not be mentioned in this chapter. Interviewees are only referred to by their job title as listed above. All of the information that is presented in this chapter was obtained from the four interviews.

A. BACKGROUND ON PROJECT X

Project X, the subject of this case study, is a software development project that is currently in the development stage. The mission of Project X is to receive and prepare critical DoD data received from a myriad of worldwide sources for storage in a centralized database and then disseminate portions of this stored data to specified DoD users. Project X performs this mission as a critical subsystem of its parent Support and Information System. The mission of Project X is included in the general mission of its parent Support and Information System. Project X is a critical component of its parent Support and Information System which is why it is considered a mission critical computer resource as defined in MIL-HDBK-387. [Ref. 31]

Alpha Organization is responsible for providing critical information support to other specific DoD organizations. Project X, as part of its parent Information Support System, is crucial to Alpha Organization's ability to provide this critical information support. Alpha Organization is responsible for designing, developing, implementing, and maintaining Project X and its parent system. Additionally, Alpha Organization also develops and maintains other organic systems and many small software applications relating to its central mission of providing information support to DoD user activities. External DoD user activities use these small software applications with their own organic hardware to receive requested information from Alpha Organization. Periodic updates to these small software applications and hardware are required due to the changing nature of Alpha Organization's mission.

Alpha Organization does not engage in software development projects for use in systems outside the realm of its mission. "Outsourced" software development support is provided to Alpha Organization in some key areas by certain Government organizations. Thus, Alpha Organization only develops software for internal use and for external users who utilize the organization's information resources. [Ref. 31]

There are two types of personnel within Alpha Organization: scientists who conduct research and develop theoretical models, and software developers who transform the research and models into information support systems. The scientists are constantly striving to find a better theoretical model that will provide more accurate scientific information (within Alpha Organization's scientific mission field) to DoD user activities. The software developers are responsible for developing and integrating the software with procured hardware into a new system so that the information developed from these theoretical models can be provided to the DoD user activities. In many cases, the scientists are "dual-hatted" as software developers.

Assignments of personnel to specific software develop projects within Alpha Organization are accomplished via a quasi-matrix system. Personnel from specific functional areas within the organization are assigned to projects based on availability, knowledge, and experience in the project subject area. Many times, the availability of personnel has taken precedence over knowledge and experience as criteria for assignment to a particular project. This is mostly due to a shortage of qualified software development personnel within Alpha Organization. Project X was profoundly impacted by this shortage of qualified software development personnel. There were no personnel available in the organization for assignment to Project X who had education and experience in the Ada programming language, the CASE tool assigned to Project X, and in the use of software reuse. This lack of experience and

education of Alpha Organization personnel within these three development requirements will be discussed further in the next sections. [Refs. 27 and 28]

Project X began development in June 1993 with the assignment of the Project Manager and six other software development personnel. Currently, along with the Project Manager, there are eight personnel assigned to the project. The completion date at the start of the project was estimated to be in October 1996. The new completion date, according to the Project Manager, is estimated to be in November 1996. [Ref. 28]

At the start of Project X, as previously stated, there were three development requirements established by Alpha Organization's upper level management which had a noticeable impact on the project. The first requirement stated that Project X software developers would use a particular CASE tool that will now be referred to as Delta CASE tool. The second requirement established FORTRAN as the project programming language. Finally, the third requirement directed development personnel to design and develop Project X using reusable software components. Six months after the project began, the FORTRAN programming language requirement was changed to Ada. The other two requirements remained the same during the change from FORTRAN to Ada. These three requirements and their impact on Project X are the focus of this thesis. We will now provide a historical look at these three Project X development requirements. [Ref. 28]

1. The Requirement to Use the Delta Case Tool

The Delta CASE tool was introduced into Alpha Organization's software development program to increase productivity and efficiency of software development. Five years ago, management was pursuing a period of system rebuilding and new project development when they considered introducing CASE tool technology. Management also wanted to use a specified software engineering

methodology along with CASE tool technology as a necessary part of the organization's software development methodology. Through the use of a designated software engineering process, the CASE tool was supposed to provide greater structure and standardization in the design of the project. It was not supposed to directly improve the development process but only help automate the process thereby increasing productivity. Says the Software Coordinator, "we wanted to give programmers the tools to do more requirements and design analysis to replace the current seat of the pants programming methods." All design information that would be stored as a result of the CASE tool was to be used for the development of future projects (reuse) and for the maintenance of internally developed software systems. The Software Coordinator further states, "these goals were thought to be very much within reach." To achieve these goals, management had produced a well thought out implementation plan for introducing the CASE tool into the project. [Ref. 27]

After establishing the CASE tool requirements and developing the implementation plan, the Software Coordinator then conducted a thorough market survey of available commercial CASE tools. After the market survey, the Software Coordinator submitted a contract request for proposal (RFP) through the proper procurement channels. Two years later, the Delta CASE tool arrived at the organization. [Ref. 27]

In spite of the substantial pre-CASE tool planning, the procurement process for the CASE tool produced substantial problems in introducing the CASE tool into the project. In the RFP, one of the requirements specified a CASE tool environment that would operate throughout the full software development process to the end result of developing Ada code. There were no commercial tools available that could achieve this requirement. Another requirement specified that the tool must operate with the disk operating system (DOS). A DOS version of the tool was available, however, the

tool was scheduled to be upgraded with the OS/2 operating system. The Delta CASE tool contract was for five years. To prevent the chance of the tool becoming obsolete within the five-year contract, the Software Coordinator chose to get the greater capability associated with the OS/2 upgrade. This decision caused considerable problems for the software developers because now they had to learn to use the OS/2 operating system along with the software engineering methods and the CASE tool in the software development project. A third requirement requested on site technical and training support for software development personnel using the CASE tool. This type of support could not be procured in the original contract. As a result, a new contract had to be developed which ultimately pushed the arrival date of the training consultant almost one year past the arrival of the CASE tool. Software developers did receive a two-week training course on the Delta CASE tool after it first arrived, but the tool was too complex to be taught in such a short time frame. [Ref. 27]

The Delta CASE tool had two specific risks associated with its introduction into Alpha Organization. The first risk was the introduction of the Delta CASE tool and Software engineering methodologies at the same time. Says the Software Coordinator, "the literature on CASE tool use warned that it was high risk to bring in the methods and the CASE tool at the same time, however, we did that anyway." The correct way would have been to introduce the methods first and then the CASE tool second. The Software Coordinator thought that introducing the methods first would "turn people off so much to the mundane amounts of paperwork that they would then be turned off to the methods." Thus, she opted to bring both the methodology and the tool in together. [Ref. 27]

The second risk related to the complexity of the Delta CASE tool. Most software developers in Alpha Organization only had limited experience with simple CASE tools. Alpha Organization's upper level management anticipated that software

developers would be able and willing to learn to use the Delta CASE tool. Despite these risks, the Software Coordinator went forward with the plan. [Ref. 27]

Three pilot projects used the Delta CASE tool before it was introduced into Project X. The first project was a small software development project used to test some of the tool's capabilities. This project was not finished. Two other small projects were also started and not completed. All pilot projects were not finished because the assigned development personnel had problems operating the Delta CASE tool. Despite these operating problems and an initial resistance from software developers regarding the tool, the Software Coordinator decided to introduce the Delta CASE tool into Project X. [Refs. 27 and 29]

2. The Requirement to Use the Ada Programming Language

The decision by Alpha Organization's upper level management to change from FORTRAN to Ada in Project X was not made in a hasty manner. The selection of the programming language for Project X was considered to be of paramount importance because the organization's upper level management wanted a language that would provide the greatest software development benefits in design, development, quality, and maintenance. FORTRAN was initially chosen because the organization had an extensive FORTRAN software reuse library, the organization had qualified FORTRAN programmers available, and the system which Project X replaces was developed in FORTRAN. However, the DoD Ada mandate stated that Ada would be used for all new projects except where not cost effective. Any deviation from the Ada mandate required a waiver approval from DoD via the DoN. Alpha Organization's upper level management considered Ada a good language because it implements and enforces software engineering practices, and the maintenance of software developed in Ada is supposed to be easier than other languages [Ref. 31]. Upper level management also saw Project X as a stepping stone to initiating a program to

gradually bring in Ada as the main programming language for Alpha Organization. Because of these attributes and reasons, upper level management chose not to request a waiver, and thus the programming language was switched from FORTRAN to Ada. [Ref. 27]

Alpha Organization's upper level management is deliberately moving slow in introducing Ada into the organization because of an industry lack of Ada tool vendor support, a lack of qualified Ada programmers, a lack of educational institutions offering an Ada curriculum, and a lack of a broad user base. The lack of industry support in these critical areas would have made a rapid transition to Ada very risky and very expensive in terms of investment costs. If the organization had decided to fully implement Ada into the organization and then DoD decides to remove the Ada mandate for lack of industry support, then the developing organization would have to make a costly transition to another programming language. By making a gradual transition to Ada, the organization is better prepared for DoD and industry changes in programming languages. [Ref. 27]

The lack of qualified Ada programmers was a tremendous problem for the PM in implementing Ada into the project. All personnel assigned to Project X received a two-week training class on the Ada language. However, this training was insufficient for personnel to adequately proceed in the development of the project. There was only one qualified Ada programmer in the organization (Programmer One) and she was assigned to Project X on a temporary loan. The organization tried to compensate for the shortage of Ada programmers within the project and the organization by contracting out for qualified Ada programmers. Contract support Ada programmers were very difficult to procure. Even after they arrived, they were either not qualified in Ada and qualified in software engineering, or qualified in Ada and not qualified in software engineering. These contract personnel had to receive additional

training before they could even be considered contributors to Project X. These implementation problems pushed the development schedule back substantially. However, the shortage of Ada programmers has now been resolved for the most part, and the project is able to proceed. [Ref. 28]

There was a general feeling in the organization that the DoD Ada mandate was established too soon. Says the Software Coordinator, "timing of the mandate was premature, because the language was not mature." In the beginning, programmers had a difficult time crossing over to different user interfaces with Ada. In spite of these initial difficulties, the Software Coordinator still considers Ada the best choice for Project X. [Ref. 27]

Project X is the only project in development within Alpha Organization that is being written in Ada. Two other smaller projects written in Ada, that are not related to Project X, have been completed. The organization has received permission from DoN to waive the Ada requirements for certain projects. Because of this waiver, Alpha Organization is able to develop other subsystems within the parent Information Support System using the C and FORTRAN programming languages. Thus, the various subsystems of the parent Information Support System are being written in multiple programming languages. [Ref. 27]

3. The Requirement for Reusable Software in Project X

Maximizing the development and use of reusable software components was a second requirement for Project X. Currently, Alpha Organization has a deliberate software reuse program in place. However, the organization has not always embraced the deliberate design and subsequent use of reusable components in systems development; the cornerstones of a deliberate software reuse program. Ten years ago, in the early development stages of the reuse program, says the Alpha Organization Software Coordinator, "components were designed for reuse only if the already

designed system would benefit or would allow the use of reusable components." Complete systems were not deliberately designed under the auspice of using and developing reusable software components. [Ref. 27]

The current reuse program of Alpha Organization consists of a reuse committee that manages and promotes software reuse within all development projects. This committee is chaired by a Reuse Coordinator and is staffed, among others, by an Ada Specialist. These two individuals are responsible for ensuring the usability of the software reuse library for project development personnel. Upon requests from Ada software developers (most are assigned to project X) for an Ada component with particular parameters, the Reuse Coordinator and Ada Specialist search various external libraries to include DoD centralized Ada repositories, to obtain the desired component. If the component is found to be of sufficient quality and is properly documented, then it is placed in the organization's reuse library. [Ref. 27]

The types or categories of software reuse employed by Alpha Organization are not as sophisticated as some employed in other successful programs in industry. Software reuse in the organization is accomplished mostly by intention in the form of black box reuse, and by product where the product most reused in the organization is source code. The goal in developing these components is to try to standardize them for many different applications. Reusing the same component many times is one of the main advantages for software reuse in Project X and for other projects in the organization. Says the Software Manager on software reuse, "it is the only way we will survive in the future because of smaller DoD budgets and an increased emphasis on reducing development costs." [Ref. 27]

The culture within the organization, according to the Software Coordinator, seems to embrace the reuse of software. However, the management (represented by the Software Coordinator) viewpoint and the views of the software developers are

somewhat different. Management wants software developers to focus more on the deliberate design of systems with reuse as a requirement. They want personnel to deliberately design components that can be used in current systems and future systems. Software development personnel, in contrast, utilize software reuse more from an ad hoc basis. Most components that are reused are either obtained from a fellow programmer or used from a previous project. Personnel are resistant to incorporate newly designed components into the centralized library because the time requirements and software process requirements are too time consuming. [Ref. 27]

Difficulties incorporating reusable software, says the Project Manager, "are the reasons developing personnel are reluctant to go through the reuse implementation process." The process for incorporating new components in the library requires a reuse committee review on quality and proper documentation for reuse components. The Software Coordinator views this process as necessary to ensure the reuse component has met the quality standards and contains a sufficient amount of documentation to aid retrieval for future reuse. Alpha Organization's Upper Level Management supports this policy, however, development schedule pressure passed down to Project X seems to push reuse development farther down on project developers' list of priorities. Developing personnel are usually constrained by schedule deadlines and consequently they do not have the time to go through the reuse implementation process. [Ref. 27]

B. BACKGROUND ON INTERVIEWEES

There were two criteria for selecting the four interviewees for this case study; (1) they had direct knowledge of the three development requirements and the requirements' impact on Project X, and (2) they were available for interviewing. In this section, we will present background information on all four interviewees. We

will also provide points of agreement and points of disagreement among the four interviewee results.

1. Personal Information on Interviewees

This section will provide relevant information on all four interviewees. Interviewee One was the Software Coordinator for Alpha Organization. The Software Coordinator provided an upper level management perspective on the three requirements and their impact on Project X. Almost all policies concerning software development were devised, developed, and introduced through the Software Coordinator. Interviewee Two was the Project Manager for Project X. The Project Manager had a perspective of policy implementor concerning the three development requirements of the project. Interviewees' Three and Four were software programmers assigned to Project X. Programmer One, Interviewee Three, was the first trained Ada programmer to work on Project X because no other Ada programmers were available in Alpha Organization. Interviewee Three was also the first programmer assigned to Project X who was required to use all three development requirements. The second programmer, Interviewee Four, was assigned to Project X a few months after Interviewee Three. Unlike Interviewee Three, Interviewee Four had no training in Ada or CASE tools prior to working on Project X.

The four interviewees have varied levels of education and experience in software development. The Software Coordinator has the most extensive combination of experience and education among the four interviewees, in the production and management of software development. In the area of formal education, the Software Coordinator has a four-year degree in Mathematics and an advanced degree in Information Technology Management. Her work experience comprises 29 years of developing, maintaining, and managing software development. The Software Coordinator completed extensive research in software reuse and then

designed the deliberate software reuse program at Alpha Organization. Although her education and experience are extensive, the Software Coordinator has very little experience in the Ada programming language, software engineering principles, or the operation of complex CASE tools. [Ref. 27]

Interviewee Two, the Project Manager, does not have quite the extensive background and education levels as does the Software Coordinator. Formal education includes a four-year degree in Computer Science. Practical experience includes 10 years in software development. Prior to his time in software development, the Project Manager was a scientist in the organization's scientific discipline. In his Computer Science degree, the Project Manager learned software engineering principles and the Ada programming language. He also has some experience and education in deliberate software reuse, but only limited experience on simple CASE tools. The Project Manager considers himself an advocate of Ada, deliberate software reuse, and software engineering, but not the Delta CASE tool. [Ref. 28]

Compared to the other three interviewees, Programmer One has more formal education, but less experience in software development. Formal education includes a four-year degree in Mathematics, an MBA, and an advanced degree in Computer Science. Her total work experience includes six years in software development. She has received several classes on software engineering principles, the use of CASE tool technology, and the Ada programming language. Programmer One also attended the two-week course on the Delta CASE tool. She was one of two programmers assigned to the second and third pilot projects which used the Delta CASE tool. In the area of software reuse, she has received some formal training, but has little work experience in deliberate software reuse. Like the Project Manager, Programmer One considers herself an advocate of Ada, deliberate software reuse, and software engineering principles, but not the Delta CASE tool. [Ref. 29]

Programmer Two, Interviewee Four, has more experience but less formal education than the first programmer. She holds a four-year degree in Computer Science and has 15 years work experience in software development. Her formal training does not include any classes in software engineering principles, the Ada programming language, or deliberate software reuse. However, she has grasped the rudiments of these subject areas through self-learning. Programmer Two is more open to change than Programmer One. She was more apt to "make the best of a bad situation" such as with the Delta CASE tool. Programmer Two considers herself an advocate of Ada, the Delta CASE tool, deliberate software reuse, and software engineering principles. [Ref. 30]

2. Points of Agreement (PoAs) Among the Interviewees

This part of the interviewee background section addresses all points of agreement among the four interviewees. Finding PoAs among the four interviewees establishes credibility in the agreed upon subject areas and allows greater ease in analyzing and interpreting these results. All PoAs are addressed in three subsections as they relate to the selected three Project X development requirements, and the primary and subsidiary research question.

a. PoAs Concerning the Delta Case Tool

- When the Delta CASE tool first arrived, all four interviewees thought it would increase productivity and decrease software development cost and time over the software's full life cycle.

- All interviewees agree that introducing the tool and software engineering methods at the same time was a mistake. Implementing both together was overwhelming for Project X software developers.

- The Project Manager and the two programmers agree that the two-week training course given to Project X personnel and other Alpha Organization personnel was inadequate. Personnel stated the course was presented too fast and the instructor was not very knowledgeable

in the Delta CASE tool. Additionally, there was inadequate technical support given to users of the CASE tool. Feedback support was not available for personnel who had questions on the use of the tool. The Software Coordinator had no opinion in this area because she did not attend the course.

- The interviewees all agree that the tool was inefficient because it required the OS/2 operating system, it could not create Ada code from Project X software designs, and it could not be used simultaneously by multiple users.

- All interviewees agree that the culture of Alpha Organization does not readily accept change. Development personnel did not accept the tool because it was too difficult to understand and they were not willing to learn or change.

- The interviewees agree that management support for the tool was lacking at all levels within Alpha Organization. There was no champion for the tool in management. Management was more concerned with personnel meeting deadlines than learning the tool.

- Both programmers agree that because of proprietary design of the tool, products designed with the Delta CASE tool could not be imported over to other tools. The output products of the CASE tool were not standardized. There was no global usage of data. This could significantly hamper software maintenance activities for this project and future software development projects.

b. PoAs Concerning the Use of Ada

- All interviewees agree that Ada was mandated for Project X, but they also agree that Ada was the best programming language for Project X. Investment costs for compilers, personnel training, and longer development times are higher than FORTRAN, but overall life cycle costs are lower.

- The interviewees agree that Project X development productivity suffered at first because personnel were not familiar with the language.

As confidence and familiarity with the language grew, so did the development productivity.

- The interviewees agree that software written in Ada is cheaper to maintain. It forces programmers to use software engineering methods.

- All interviewees agree that the strong typing and generic packaging characteristics of Ada are good for designing reusable components.

- The interviewees agree that Alpha Organization personnel outside of Project X do not embrace the change to Ada. Most personnel outside of Project X are still ingrained in using FORTRAN.

- All interviewees agree that the shortage of qualified Ada programmers in Project X decreased productivity. They also stated that this shortage is a DoD and industry wide problem.

- Both the Software Coordinator and the Project Manager agree that productivity suffered because of the shortage of qualified Ada programers. Available personnel had to be trained in Ada. Consequently, their productivity suffered until they became proficient using the language.

- Both the Software Coordinator and the Project Manager agree that Ada compilers were substantially more expensive than FORTRAN or C compilers. An Ada compiler costs between $5,000 and $10,000. This is compared to the $100 cost of a C or FORTRAN compiler. The costs are even greater with a multi-use license. A multi-use license for five people was required for Project X and it cost $50,000. Compare this to the $300 cost for a multi-user license for a C or FORTRAN compiler.

c. PoAs Concerning Deliberate Software Reuse

- All interviewees agree that Alpha Organization has a deliberate formal software reuse program.

- The interviewees agree that management within Alpha Organization supports reuse, but management does not enforce the design and

construction of reusable components. When schedule pressures arise, management is more interested in designing systems and components rapidly rather than investing time and effort into developing reusable components and systems.

- According to all interviewees, the culture at Alpha Organization is resistant to endure the initial "pains" of increased development times and costs associated with designing systems comprised primarily of reusable components.

- The interviewees agree that software developers in Project X use more black box (unmodified) source code reuse.

- All interviewees agree that the organization's Ada reuse library is small and requires more modules to become beneficial to developers.

- The interviewees agree that all personnel within Alpha Organization know the procedures for placing software modules in the organization's central Ada reuse library. They also agree that the process requirement of placing modules in the library is too time consuming, requiring strict adherence to quality standards and proper documentation requirements.

- The two programmers agree that Project X developers are using some type of software reuse. However, most of the project's development personnel are engaging in ad hoc reuse; they informally share components with each other.

- All interviewees say that software reuse in general, lowers future development costs, requires less time to develop future components and systems, and results in higher quality software components.

- All interviewees agree that increasing the scope of the organization's reuse program to include reusing designs and architectures would be beneficial. However, in accomplishing this task, the interviewees agree that developing reusable components must be given a higher priority over schedule deadlines, and personnel must be more willing to accept the process of placing components into the organization's central Ada reuse library.

- The Software Coordinator and the Project Manager agree that development costs for reusable components in Project X are higher due to the increase in time required to build the reusable components. Reusable components require greater emphasis on component standardization, quality, and proper documentation. Greater component standardization requires more collaboration among developers in designing a component that can be used in several areas now and in the future. [Refs. 27 and 28]

3. Points of Disagreement (PoDs) Among the Interviewees

This part of the interviewee background section addresses all points of disagreement among the four interviewees. The results of the interviews show there are far fewer PoDs than PoAs. Similar to the PoAs, the PoDs are addressed as they relate to the selected three Project X development requirements, and the primary and subsidiary research questions.

a. PoDs Concerning the Delta CASE Tool

- Programmer One and the Project Manager did not like the tool, while Programmer Two considered the tool useful. This was partially due to differences in attitude between the two programmers regarding the tool.

- Programmer One was trying to use the tool for the full development process, whereas Programmer Two only used the tool for simple design functions such as basic diagram construction.

- Programmer One said the Delta CASE tool increased her development productivity where Programmer Two stated that the tool decreased her productivity by three fold. The Project Manager also stated that development productivity decreased because of the tool.

- Programmer Two was able to find ways to work around the shortcomings of the Delta CASE tool. Programmer One became discouraged with the shortcomings of the tool.

b. PoDs Concerning the Use of Ada

There were no PoDs among the four interviewees concerning the use of the Ada programming language in Project X. In the interviews, some interviewees gave more information in a particular area of Ada use than others. However, in general, the interviewees responded similarly to interview questions concerning Ada use in the project.

c. PoDs Concerning Deliberate Software Reuse

- The Project Manager says that Project X uses vertical, planned, black-box source code reuse. Both programmers say that reuse is not planned and mostly consists of ad hoc, black-box source code reuse.

- Both programmers state that most development personnel do not know how to build reusable components that can be used in Project X and in future systems. The Project Manager says that Project X development personnel do know how to design reusable components.
- Programmer Two says that 50% or more of Project X will consist of reusable components, whereas the Project Manager stated that 80% of the project will consist of reusable components.

This section provided personal background information on the four case study interviewees: the Software Coordinator, the Project Manager, Programmer One, and Programmer Two. Additionally, PoAs and PoDs among the four interviewees were also provided in this section. The four interviewees had varying levels of formal education and experience in software development. Experience in software development ranged from 29 years for the Software Coordinator to six years for Programmer One. Programmer One had the most extensive formal education of the four interviewees. The interviewees each held different points of view in relation to the development requirements. However, there were far more PoAs among all the

interviewees concerning the development requirements than there were PoDs. Greater agreement among the interviewees concerning the three development requirements establishes greater credibility for the interview results.

C. IMPACT OF THE DELTA CASE TOOL

This section provides relevant results on the impact of the Delta CASE tool that were not covered in the PoA and PoD sections. Similar to the last sections, information is presented in a summarized format. As stated before, the focus of the interview questions and the subsequent responses to those questions is more on management issues vice technical issues. Information from the interviews pertaining to technical issues is only provided if it relates to the primary and subsidiary research questions.

1. Interview Results from the Software Coordinator

- Bringing in methods and tools together produced problems for the Project X software developers. The developers had a difficult time with the methods and they blamed this on the CASE tool. According to the Software Coordinator, "the tool was telling the developers that what they were doing was wrong, but the developers saw this as a flaw or weakness in the tool." In reality, the tool can help guide you through the method, but it does not help improve the learning of the method. The consequence of this problem was that the developers became dissatisfied with the tool. Developers blamed the tool for the development problems and the decrease in productivity. [Ref. 27]

- The Software Coordinator overestimated the Project X developers' desire to learn to use the CASE tool and accept the decrease in productivity that follows the introduction of the tool. Project X development personnel thought that the tool would do more in the development process than it actually did. They were still trying to use the old traditional methods with the tool. As a consequence, development productivity decreased and the project slipped more behind schedule. Management should have started with a simpler tool

and then add capabilities to it as personnel grew to accept the tool in the development process. [Ref. 27]

- The Software Coordinator attributed the dissatisfaction with the Delta CASE tool among Project X personnel to a lack of software engineering experience. [Ref. 27]

- Project X development personnel struggled with the Delta CASE tool following the initial substandard two week training session. They became disgruntled and blamed the tool for their decrease in productivity. Even after the arrival of the full time tool consultant, most developers were still not able to effectively use the tool. Development personnel lost faith in the tool's capabilities and were unwilling to use it in the development process. [Ref. 27]

- The tremendous gap between the time the tool was first contracted and when it arrived made the tool obsolete. By the time the tool did arrive, project requirements had changed (programming language had changed from FORTRAN to Ada) and there were better tools commercially available. [Ref. 27]

2. Interview Results from the Project Manager

- Since the tool was not designed for Ada, Project X developers had a difficult time using the tool for any development tasks other than within two simple functional areas. Says the PM, "the tool was difficult to understand, learn, manipulate, and use." [Ref. 28]

- The Project Manager placed partial blame on the procurement of the unsuccessful Delta CASE tool to his lack of input in the selection process. The Project Manager had no voice in how or what CASE tool was selected. He also had no say as to how the tool was introduced into the project. [Ref. 28]

- The Delta CASE tool cost Alpha Organization several hundred thousand dollars over what should have been required. Extra costs for the tool were attributed to the requirement for additional OS/2 work

stations, additional training classes on OS/2, and the one year contract for the Delta CASE tool consultant. [Ref. 28]

- The difficulty with the tool amplified by the additional learning requirements further reduced the productivity of the project developers and pushed the project further behind schedule. The PM was granted a waiver from using the CASE tool because of this reduction in productivity. [Ref. 28]

- Project X development personnel experienced training problems with the Delta CASE tool. Personnel received the two-week training class, however, they did not start using the tool until nine months later. Following the class, limited technical support (CASE tool vendor personnel) was periodically available to help personnel transition to using the tool. However, the quality of the vendor personnel was questionable (they lacked knowledge in systems and software engineering) and their turnover was high which further frustrated the project developers when trying to work with the tool. By the time the permanent Delta CASE tool consultant had arrived, negative attitudes toward the tool were fairly concrete. [Ref. 28]

3. Interview Results from Programmer One

- The root cause for the difficulties in using the Delta CASE tool according to Programmer One, was the CASE tool itself. The CASE tool was not designed well and was not right for the projects. [Ref. 29]

- Because of the problems experienced with the CASE tool, Programmer One decided not to use this tool for Project X. Instead, Programmer One used a simpler tool in developing the project. The Project Manager approved this decision. [Ref. 29]

- The CASE tool was too difficult to use. There was no consistency in its use. The same task executed more than one time on the CASE tool resulted in different outputs each time. Programmer One blamed this problem on "bugs" in the CASE tool software. Because of this

problem, Programmer One and other programmers lost faith in the tool. [Ref. 29]

- Many users of the CASE tool thought that by using the tool, they were doing software engineering. This was a learning or perception problem caused by the introduction of software engineering methods along with the CASE tool. Personnel blamed software engineering methods for problems caused by the CASE tool. [Ref. 29]

- Programmer One stated that a better tool could have been found to increase productivity; a tool that was more user friendly and was designed to be used with general software engineering methods. [Ref. 29]

4. Interview Results from Programmer Two

- Programmer Two was not exposed to any of the more complex areas of the tool. [Ref. 30]

- Programmer Two did not have enough experience with CASE tools to know if there was a better tool available then the Delta CASE tool. [Ref. 30]

This section identified all interview results on the Delta CASE tool not provided in the last section on PoAs and PoDs. The Software Coordinator put partial blame on both the development personnel and the Delta CASE tool for the tool's ultimate failure in Project X. In contrast, the Project Manager and Programmer One blame the tool's complexity, inability to generate Ada code, and lack of standards as reasons for its failure in Project X. Programmer One was the only true advocate of the Delta CASE tool among the four interviewees. The next section provides additional interview results on the use of Ada in Project X.

D. IMPACT OF THE ADA PROGRAMMING LANGUAGE

This section provides information obtained from the four interviews that pertains to the impact of the Ada programming language. All information presented in this section are interview results that were not provided in sections A and B. The organization of this section will be similar to the last section where the interview responses of the Software Coordinator, Project Manager, and both programmers will be presented as separate subsections.

1. Interview Results from the Software Coordinator

- The ICASE tool, an integrated CASE tool that DoD is developing, will be able to support the full software development process to include Ada code generation. [Ref. 27]

- According to the Software Coordinator, "there are higher investment costs with Ada but life cycle costs are cheaper." The investment costs of obtaining and training personnel, tools (if even available), and compilers are very high. However, because of the inherent capabilities of Ada and the fact that it is a standardized language, Ada components developed in Project X are cheaper to maintain and have greater capability for reuse. [Ref. 27]

- Ada is considered by the Software Coordinator to be a good decision for Project X. However, the decision to use Ada in future projects will be contingent on whether vendor support and market share of Ada use will grow in the future. The fate of Ada is still unknown at this time. [Ref. 27]

2. Interview Results from the Project Manager

- The Ada language is much more structured than FORTRAN. Long term maintenance for Project X will be easier and will cost less. [Ref. 27]

- Within Project X, the Project Manager had problems interfacing to applications written in languages other than Ada. The problem still exists, however, it would have been more severe had FORTRAN been the language. Ada 95 is supposed to help solve this problem.

- The Ada programming language provides self-documenting capabilities in Project X. [Ref. 28]

- Ada use in Project X, according to the PM, is promising lower life cycle costs because the quality of the components is higher and the maintenance costs are lower as compared to the other subsystems within the parent Information Support System. Many of the other subsystems are being written in other programming languages such as C and FORTRAN. These languages are more prone to higher maintenance than Ada. Says the PM, "there could be a huge money drain on software maintenance because of the piecemeal design of the parent system." Upper level management is willing to "eat" the additional maintenance costs rather than make a total commitment to Ada for the entire project. They are not willing to risk moving entirely to Ada until the language establishes greater acceptance in the commercial industry. [Ref. 28]

3. Interview Results from Programmer One

- The Ada language was difficult to use in the beginning. Says Programmer One, "Ada was difficult to use at first because the original designs were based on FORTRAN." Developers learned to redesign using Ada principles/concepts. Development productivity was slower as compared to FORTRAN, but Ada will increase productivity through easier maintenance of modules and through the reuse of components in future projects. [Ref. 29]

4. Interview Results from Programmer Two

- Most of the components used in the development of Project X are developed by the project personnel. Very few previously developed Ada components were available for reuse in the project. The development of reusable components for Project X will benefit future development projects. [Ref. 30]

- Programmer Two blames DoD's past use of multiple languages on the shortage of Ada programmers in Project X. Even the good programmers that are available are very expensive to obtain. The shortage of qualified programmers is partly to blame for the schedule slip following the introduction of Ada into Project X. [Ref. 30]

This section identified all interview results relating to the Ada requirement not provided in the section on PoAs and PoDs. The decision to use Ada in Project X was mandated by DoD. However, all four interviewees agree that Ada was the best choice for Project X. Ada has higher investment costs, but these costs are offset by increased quality, lower maintenance, and greater potential for reuse. The next section identifies additional interview results on software reuse in Project X.

E. IMPACT OF THE SOFTWARE REUSE PROGRAM

This section provides additional information obtained from the four interviews that pertains to the impact of software reuse on Project X. Like the last section, the responses of the Software Coordinator, Project Manager, Programmer One, and Programmer Two are presented as separate subsections within this chapter. This section presents information not previously covered in Section A and B. Only information required to support the primary and subsidiary research questions is presented so as to keep within the scope of the thesis.

1. Interview Results from the Software Coordinator

- The benefits of software reuse development for Project X and other future projects in the organization is lower maintenance costs and lower future development costs. Software reuse, according to the Software Coordinator "is definitely cost effective," because the higher development costs are paid for through future reuse of components. [Ref. 27]

2. Interview Results from the Project Manager

- More time and effort have been required to develop reusable components for Project X because of the substandard initial quality of Ada programmers. [Ref. 28]

3. Interview Results from Programmer One

- Programmer One has modified (white box) several Ada components for reuse within Project X. This action still equates to ad hoc reuse because components are not specifically and deliberately designed to be reused within Project X and in future projects. [Ref. 29]

- Most personnel know that reuse saves future project development dollars. However, because personnel have insufficient skills in software engineering principles and the Ada language, they are not able to adequately design and build reusable components for different system applications. [Ref. 29]

- Programmer One is not sure whether software reuse for this project has increased productivity because of the learning problems with Ada, software engineering principles, and the project CASE tool. There is no way of measuring differences in levels of productivity, concerning maintenance time and cost for the project because no metrics were established and the project is still in development. She said it is difficult to assess whether the lower maintenance costs and potential for

use in other systems will increase the life cycle productivity of developing reusable components.

- Programmer One says software developers in Alpha Organization are highly encouraged by management to develop and use reusable components. However, she thinks that management is not aware of what is actually going on in the area of software reuse. She has viewed more ad hoc, opportunistic reuse of software components informally between individual programmers. Personnel are reluctant to go through the component reuse incorporation process because of time constraints, and the strict quality and documentation requirements. Management is not aware of what is going on because leadership and oversight management are lacking. [Ref. 29]

4. Interview Results from Programmer Two

- Programmer Two says there is more "lip service" given to the organization's formal reuse program by development personnel than actual reusable component design. [Ref. 30]

- The organization has a very large FORTRAN reuse library. Most software developers in the organization are very happy with FORTRAN. In comparison to the FORTRAN library, the Ada reuse library is very small. Says Programmer Two, "the Ada library needs a lot of improvement." However, most personnel are still too ingrained in using FORTRAN. The only personnel developing any significant software in Ada are the personnel assigned to Project X. Programmer Two blames both management and development personnel for this lack of support for the Ada reuse library and the Ada language in general. [Ref. 30]

This section provided interview results on Project X software reuse not found in the section on PoAs and PoDs. Alpha Organization management has high expectations for software reuse within Project X. However, according to both programmers, most development personnel are still designing reusable components

using ad hoc opportunistic practices. Management does not seem to be aware of the magnitude of this problem.

F. SUMMARY

Information presented in this chapter represented the results of four interviews with personnel associated with Project X. Questions asked in the interviews are listed in Appendix A. There were two criteria for selecting the four interviewees for this case study; (1) they had direct knowledge of the three development requirements and the requirements' impact on Project X, and (2) they were available for interviewing. The four interviewees selected were the Software Coordinator, the Project Manager, and two Project X programmers.

All information obtained from the four interviews related to three specific development requirements of Project X: the Delta CASE tool, the use of the Ada programming language, and the use of software reuse. Section A of the chapter provided background information on Alpha Organization, Project X, and the three specified development requirements. Section B presented background information on the four interviewees to include formal education, software development experience, and work history in Alpha Organization and Project X. Also presented in Section B were PoAs and PoDs among the four interviewees concerning the three development requirements. Sections C, D, and E presented additional interviewee information on the impact of the three development requirements on Project X. The next chapter, Chapter V, will use these results for further analysis and interpretation.

V. ANALYSIS AND INTERPRETATION OF RESULTS

The purpose of this chapter is to analyze the results of the interviews, apply those results to the literature found in Chapter II, and provide additional interpretation of those results. Section A consists of the analysis portion where pertinent interview results are compared and analyzed with the literature presented in Chapter II. Section B provides an interpretation of the information presented in Section A. All interpretations of the case study information is strictly the opinion of the researcher. The material presented in Sections A and B directly applies to the three specific development requirements of this case study and their effect on Project X. Furthermore, the information presented in this chapter will be used in Chapter VI to answer the primary and subsidiary research questions. Finally, Section C of this chapter provides a summary of the analysis and interpretation sections.

A. ANALYSIS OF INTERVIEW RESULTS

This section analyzes the results of the four interviews on the three development requirements of Project X. The analysis consists of comparing the interview results with information found in the literature review. Comparing the interview results with published industry literature provides a baseline from which the researcher can analyze (1) the effectiveness of Alpha Organization's strategy and implementation plan concerning the three development requirements and (2) the impact of the three development requirements on Project X. Information on the three development requirements is presented in a logical sequence by tying together relevant facts and interviewee opinions. To better enhance the analysis, each development requirement is presented separately.

It was apparent in the interview results that Alpha Organization was going through a period of transition from "seat of the pants programming" to the more

scientific methods associated with software engineering principles. Reduced DoD budgets and greater demand for quality, "upgradeable," and maintainable software in the organization was forcing the change. The Software Coordinator had developed a vision and a strategy for incorporating change into Alpha Organization's software development processes. However, for the change strategy to work, the following two critical areas must be satisfied first: (1) future development systems' requirements (i.e., programming language and reusable software) must be consistent, and (2) development personnel / management must support the changes. The three specified development requirements of Project X are three areas of change in Alpha Organization's development philosophy. The incorporation of these three development requirements and their effect on Project X is discussed in the next three sections.

1. The Delta Case Tool on Project X

The Software Coordinator and other senior management personnel in Alpha Organization first considered CASE tool technology in order to improve their software development process. Literature supports this reasoning where organizations with a properly implemented CASE tool can achieve greater productivity and software quality with less personnel [Ref. 4]. The key to this last statement is a "properly implemented CASE tool." The implementation plan for introducing CASE tool technology into the organization and more specifically, Project X, was flawed from the beginning. In this section, the decision to procure the Delta CASE tool is analyzed for form, fit, and function within Project X. The second part of this section analyzes the effect of this decision on Project X.

a. The Decision to Use the Delta CASE Tool

There were five specific problem areas in upper level management's decision to purchase the Delta CASE tool, (1) the tool could not generate Ada code, (2) the tool and the software engineering development methods were introduced at the

same time, (3) the tool required the OS/2 operating system, (4) there was a complete lack of training and technical support for personnel using the tool, and (5) the tool was not cost effective for Alpha Organization.

The first problem area was the decision to purchase a tool that could not fulfill the requirement of generating Ada code. Like any other kind of tool, specific CASE tools can only be used in specified ways. A CASE tool that was designed for editing cannot be used for code generation. When the Software Coordinator and other senior management first began research on a CASE tool for Alpha Organization, they established a requirement for a tool that could be used throughout all phases of the development process to include code generation. During the time when CASE tools were being researched by the Software Coordinator, DoD was beginning to push the Ada language mandate. Organization management new of this mandate and decided to make a gradual change within the organization to using Ada in software development. If Alpha Organization upper level management was implementing a plan to slowly change to the Ada language, why did they choose to procure a tool that only produced COBOL code? The Software Coordinator stated that the reason the Delta CASE tool was procured was because there were no Ada code CASE tools available. This decision goes against information found in the literature which states that tools should be purchased for competitive advantage [Ref. 7]. Paying substantially more for a tool that produces unusable code does not constitute competitive advantage. If management wanted to make a gradual change to Ada or at worst was uncertain as to the language requirement for future projects, than it would have been more economical to purchase a tool that could be used in all development stages except code generation.

Introducing the tool and the software engineering development methods together was the second problem area in introducing the Delta CASE tool into Project

X. Information in the literature states that software developers must have expertise in and strictly follow one system development methodology before the introduction of a CASE tool will benefit the develop process [Ref. 9]. Most software development personnel were using traditional "seat of the pants" programming. Developers were not using any particular software engineering methodology because most personnel had only limited training in this area. When the Delta CASE tool was introduced, Project X developers were immediately required to learn the tool and the chosen methodology, and use both in developing the project. It is important to note that the culture in Alpha Organization was very resistant to change. Here, management wanted developers to use a totally new development methodology along with a new complex CASE tool that did not generate Ada code to develop project software. Developers were also required to stay on the development schedule. The result was that developers were overwhelmed. Productivity in the project suffered and personnel blamed the Delta CASE tool for the slip in productivity. Many of the developers, such as Programmer One, refused to use the tool. The end result was that schedule pressures forced upper level management to grant the Project X Project Manager a waiver from using the Delta CASE tool.

Management's decision to procure the OS/2 version of the Delta CASE tool was the third problem area. This problem area had two implications: (1) development personnel were not familiar with OS/2 and (2) there were substantially higher investment costs associated with the OS/2 version. With the introduction of OS/2, development personnel were now required to learn this new operating system, a new development methodology, and the tool itself just to be able to use the tool to develop Project X software. This new learning requirement did not generate any buy-in from development personnel regarding the Delta CASE tool. Personnel were

overwhelmed with this additional learning requirement and productivity suffered as a result.

The other implication with the OS/2 version was the substantial increase in investment costs. Since no other user systems in Alpha Organization used OS/2, new computer work stations had to be procured solely for the Delta CASE tool. Additionally, personnel had to receive training on the OS/2 operating system. This additional training increased costs in three ways: (1) the cost of the training classes themselves, (2) the costs associated with lost worker productivity during training times, and (3) the costs associated with a decrease in productivity due to developer unfamiliarity with OS/2. The final result with the decision to procure the OS/2 version was that development personnel resisted the tool even more and the organization wasted scarce budgetary resources.

A lack of available tool user training and technical support is the fourth problem area. Information in the literature states that there is a positive relationship between the degree of technical training support available and the degree of CASE tool usage [Ref. 9]. Upper level management new that the requested full time tool consultant would arrive one year after the Delta CASE tool. Yet, they still went through with the tool procurement. A two-week introductory class was presented to organization development personnel by a Delta CASE tool vendor representative, but class attendees were not able to grasp the complexities of the tool or the large volume of information in such a short time. Furthermore, during the year before the arrival of the full time consultant, sporadic tool vendor technical support was available to tool users. However, the turnover rate of these personnel was high, most of these technical advisors lacked knowledge in software engineering and systems engineering principles, and there was no consistency in the advice they gave to tool users. By the

time the full time consultant arrived, most tool users had already developed negative opinions of the tool.

The final problem area was the decision to procure the Delta CASE tool as a means of increasing the organization's software development cost effectiveness. The Software Coordinator chose the Delta CASE tool because it was viewed as a means to increase cost effectiveness in software development. However, information from the literature and the results from the interviews indicate that the tool would not be cost effective for software development in Alpha Organization. The literature from industry and business states that it is the size of the organization, the quality of it's software engineers, and the size of the products that are built that determines whether CASE tools are used and seen as a cost effective approach to developing quality software products. CASE tool use is most prevalent in large commercial organizations which are in competition with other commercial organizations. These large commercial organizations are constantly developing new software applications and view the incorporation of CASE tools as a cost effective investment that saves development costs and time through increase productivity and product quality. [Ref. 7] Alpha Organization only develops software for internal use and for organizations requesting data support. Alpha Organization is not in competition with anyone, and lacks quality software engineers. Furthermore, in the next two years following the completion of Project X and other subsystems within the parent Information Support System, software development in Alpha Organization will only consist of small software development applications and maintenance of current systems. Considering the rapid speed from which CASE tools can become obsolete and the fact that the Delta CASE tool has a five year contract, it is questionable as to whether the Delta CASE tool was a cost effective investment for Alpha Organization. Of course, this problem does not even take into account a further erosion in the tool's cost

effectiveness due to the loss in development productivity and increased investment costs.

At this point, upper level management has decided to purchase a complex CASE tool that (1) cannot be used for Ada code generation, (2) has no buy-in from the organization's development personnel, (3) requires an unfamiliar operating system with increased investment costs, and (4) will be introduced concurrently with the organization's new development methodology. The next section will analyze the effect of this decision.

b. The Effect of the Delta CASE Tool on Project X

The final effect of the Delta CASE tool on Project X was a tremendous decrease in productivity. Programmer One had the most experience with the Delta CASE tool and her testimony will have a lot of weight in this analysis. Programmer One first witnessed this decrease in productivity as a member of the development team for the last two pilot projects for the Delta CASE tool. Both pilot projects could not be completed in the scheduled time because of the complexity of the Delta CASE tool and the development personnel's reduction in productivity. By the time Programmer One was assigned to Project X, she had already struggled with the tool for several months. She continued to struggle with the tool on Project X along with other developers. The Project Manager became concerned with the lack of productivity and the development schedule. Development personnel directly attributed this reduction in productivity and schedule slip to the Delta CASE tool. The Project Manager was granted a waiver from using the tool because of the decrease in productivity. It is important to note that developer productivity has increased following the waiver and Project X is back on its original develop time schedule.

Programmer Two is the only known software developer in Alpha Organization that says the tool has increased her productivity. She is also considered to be the only software developer that is an advocate of the Delta CASE tool. It is important to note that Programmer Two only uses the simple diagram construction functions of the tool. She does not use any of the more complex functions that Programmer One and other software developers used. She also does not use the tool as was intended by the vendor. In order to use the tool in these simple functional areas, Programmer Two has to work around the tool's deficiencies. If the tool was functional, than these "work arounds" would not be required. With a better tool, Programmer Two could achieve even better productivity. Regardless of her attitude towards the tool, Programmer Two is alone among all other software developers in supporting the tool. The root causes associated with this lack of support and their effect on the productivity of Project X will be presented next.

To summarize this section, there are five root causes for Delta CASE tool's decrease in productivity within Project X. First, was the mistake of introducing the development methodology and the tool at the same time. Second, was the tool's inability to generate source code for Project X, an original capability requirement. The third cause was the procurement of the OS/2 version of the tool. Fourth, was a complete lack of quality technical and training support. Finally, the fifth cause was the sheer complexity and lack of quality associated with the tool. The organization's cultural resistance to change was strongly effected by each of these five root causes. Development personnel were positive about the tool before it arrived. However, this positive attitude turned negative quickly as personnel became overwhelmed with the requirement for change associated with these five root causes. As attitudes turned negative so too did the support for the Delta CASE tool ultimately resulting in decreased productivity for Project X.

2. The Ada Programming Language on Project X

The use of the Ada programming language in Alpha Organization was mandated by DoD. However, even without the mandate, the Software Coordinator still would have chosen to incorporate Ada into Project X. Unlike the Delta CASE tool, the introduction of the Ada programming language had a positive effect on Project X. The introduction of Ada promises higher productivity, quality, performance, and greater reusability [Ref. 3]. Project X development personnel, after their initial resistance, noticed increased productivity and greater potential for software reuse with Ada. This section provides an analysis of the interview results on Ada and compares these results with information found in the literature review. The first part of this section analyzes the decision to incorporate Ada into Project X, while the second part analyzes the effect of Ada on Project X.

a. The Decision to Use the Ada Programming Language

All the reasons why DoD chose to develop and mandate Ada use in DoD software development are the same reasons why Ada is a good choice for Project X. Like DoD organizations in the 1970s and 1980s, Alpha Organization is using several different languages in systems development [Ref. 3]. Multiple languages are currently being used to develop Project X's parent Information Support System. Like DoD systems of the past, Project X's parent system is experiencing interface problems between the different subsystems because of the incompatibility problems associated with multiple programming languages [Ref. 3]. Additionally, the use of multiple languages in Alpha Organization requires development personnel to become more specialized. Greater specialization reduces the organization's flexibility in assigning personnel to different projects. Multiple languages also requires the organization to procure more assets to support these languages such as multiple compilers, multiple reuse libraries, multiple CASE tools, and greater training

requirements. As Ada is incorporated into more of the organization's development projects, the problems with multiple languages should disappear.

The Software Coordinator, as the upper level manager, chose to slowly implement Ada into Alpha Organization's software development program. Currently, Project X is the only major development project using Ada. A slow, conservative implementation plan was chosen because of a lack of Ada tool vendor support, a lack of qualified programmers, a lack of educational institutions offering Ada curricula, and a small user base in industry. The Software Coordinator is waiting for an increase in industry's acceptance of Ada before a total commitment is made to Ada in the organization. Information in the literature supports her logic in stating that industry lacks support for Ada [Ref. 16].

Ada is also not supported by most development personnel in the organization outside of Project X. As previously stated, the organizational culture is very resistant to change. Most of the organization's development personnel have become very comfortable with FORTRAN and other languages. FORTRAN programmers, as compared to Ada programmers, also have more extensive resources available to them in the organization. If management should decide to increase Ada's use in the organization's development projects, they are likely to experience resistance from many developers. However, this resistance will probably be temporary. With the right incentives and motivation, these development personnel, like those of Project X, will learn to accept Ada.

The literature information states that Ada is the language of choice for weapon system designers, but the attitude of many commercial sector software engineers is that C++ is the only serious programming language [Ref. 11]. The popularity of C++ in the commercial sector has made it increasingly difficult for Ada to break in [Ref. 15]. The difficultly of Ada establishing a foothold in the commercial

sector has resulted in a shortage of vendors offering Ada CASE tools, Ada compilers, and other Ada products. DoD's recent introduction of the standardized Ada 95 language and a revitalized marketing strategy should increase Ada support in DoD and the commercial sector. The decision to use Ada for future projects in Alpha Organization, according to the Software Coordinator, is contingent upon whether vendor support and commercial market share of Ada use will grow in the future.

Even without the DoD mandate, the Software Coordinator still would have chosen to incorporate the language in its current limited capacity because of the following three benefits: (1) Ada enforces the use of software engineering, (2) software written in Ada is cheaper to maintain, and (3) Ada's strong typing and generic packaging characteristics make it good for designing reusable components. All three benefits result in reduced future development costs. The literature review found strong support for these three benefits. Information in the literature states that Ada supports the practice of software engineering principles, lowers software development costs, increases software quality, and lowers maintenance costs [Ref. 3].

The Software Coordinator's decision to incorporate Ada is sound, but the limited degree to which she chooses to incorporate the language is fundamentally flawed. Currently, Ada has only a limited role in the organization's software development process. Until Ada increases in popularity, Alpha Organization will continue to use the same multiple languages (plus the addition of Ada) in the development of present and future projects. Even though Ada is only used in a limited role, its use still requires specialized Ada tools, compilers, and trained software developers. The requirement for additional Ada assets and development personnel equates to higher development costs. The organization is actually spending more in investment costs by using the Ada language with the other languages. Furthermore, Alpha Organization is not reaping the benefits of having one

standardized language where software can be used and transported across the various computer environments within the organization.

The analysis indicates that the Software Coordinator's decision to use the Ada language was sound in context, but not in the degree to which the language was incorporated into the organization's software development process. Investment costs for programming language support in the organization are higher since Ada's introduction. Cost for Ada compilers alone are 50 to 166 times higher than FORTRAN Compilers [Ref. 28]. In addition to higher investment costs, the organization still has the language interface problems and resource limitations associated with using multiple languages. If the introduction of Ada coincided with the removal of other languages within Alpha Organization, than investment and support costs for programming languages would have been reduced. Higher investment costs was the one limitation the Software Coordinator was willing to forego in her conservative Ada implementation plan. [Ref. 27] The next section looks at the effect of this implementation plan and the Ada language itself on Project X.

b. The Effects of Ada on Project X

The introduction of Ada into Project X has produced some noticeable negative and positive effects on the project. Negative effects of Ada on the project include: a substantial increase in investment costs, a severe shortage of qualified Ada programmers, and an initial decrease in productivity. Positive effects of Ada include: lower life cycle development costs, support for software engineering principles, and the development of reusable Ada components.

Introducing Ada into Alpha organization and ultimately Project X has produced much higher investment costs. Information in the literature supports the premise that Ada requires higher investment costs [Ref. 3]. The problem of higher

investment costs is simple supply and demand. There are few vendors that develop compilers and CASE tools that support the Ada language. For example, Ada compilers are 50 to 166 times greater in cost when compared to a FORTRAN compiler. There is more commercial support for the FORTRAN language. More vendors are in competition for supplying FORTRAN tools and compilers to a limited demand. Until Ada develops the same commercial support, investment costs will remain high.

The severe shortage of qualified programmers in Project X had a direct causal effect on the decrease in development productivity. The shortage of qualified Ada programmers in Project X coincides with shortages in industry. When Project X was first initiated, Programmer One was the only Ada programmer available. She was assigned to the project on a temporary loan until other programmers could be found. Two separate contracts with government contractors produced less than satisfactory Ada programmers. Contract programmers were either knowledgeable in software engineering principles or the Ada language but not both. The Project Manager had to commit extra resources and time just to get the contract Ada programmers up to a level of training where they could be used in the project. Development productivity suffered initially because of the training deficiencies ultimately resulting in a development schedule slip. As personnel became more familiar with the structure of the language, development productivity increased. The project is now basically back on schedule. Information in the literature did state that productivity is slightly reduced during the first Ada project. However, if qualified Ada programmers had been available from the start of the project, it is highly unlikely that the project would have fallen behind schedule.

The analysis of the Ada language in Project X also produced some positive effects. These positive effects are as follows: (1) Ada fosters the support of

software engineering principles, (2) development personnel predict that software developed in Ada will have lower life cycle costs, (3) components developed in Ada will have greater reuse capability. All three positive effects are interrelated. The introduction of Ada complemented the introduction of software engineering principles. Development productivity increased as a result of Ada and software engineering principles. Additionally, the strong typing and generic packaging characteristics of Ada has increased the reuse potential and quality of developed components. The increased productivity, increased quality, and greater potential for reuse in Project X results in lower future development costs and lower life cycle costs. Thus, all three positive effects of the language are interrelated.

Overall, the analysis indicates that Ada has produced a positive effect on Project X. Alpha Organization did experience higher investment costs and an initial shortage of qualified Ada programmers, but their effect on the project was only temporary. As confidence and familiarity with Ada grew, development personnel began to discover increased productivity, increased quality, greater reusability, and lower costs in developing Project X software. In the next section we will analyze the effect of software reuse on Project X.

3. Software Reuse on Project X

The process of software reuse was not new to Alpha Organization when it was introduced into Project X. A software reuse program had been in place several years before the start of Project X. The result of this long standing program was an extensive FORTRAN reuse library. However, this FORTRAN library could not be used to support Project X's Ada language requirements. A new Ada library had to be built. Most of the initial reusable components for the Ada library were to come from Project X. One of Project X's development requirements specified the development and use of reusable components for the project and the Ada reuse library. Project X

development personnel supported this requirement, but in many ways they failed to execute the requirement effectively. The following areas are discussed in the analysis of software reuse in Project X: (1) the Software Coordinator's vision of success, (2) the incorporation of Ada software reuse in Project X, and (3) the effect of software reuse on Project X.

a. The Ada Reuse Vision of Success

The Software Coordinator made the decision to incorporate deliberate software reuse in Project X. A formal reuse program was already in place and personnel were trained in how to place components into the library. The Ada reuse library was small, but the Software Coordinator anticipated that the library would expand after the start of Project X. Project X, with Ada as the programming language, is to be deliberately designed and developed under the auspice of software reuse. Development personnel are to create software components that can be used in many areas within Project X and within future systems. These components are to be incorporated into the organization's Ada reuse library for present and future use.

To make software reuse more beneficial to the organization, Project X, and future development projects, the four interviewees agree that the scope of the organization's reuse program should be increased to include the reuse of designs and architectures. This increase in scope will be a step above the current practice of reusing Black-box source code. According to the literature, design and architecture reuse promises even "higher payoffs" than source code reuse [Ref. 20]. To accomplish this objective, management and development personnel both must look at software reuse from a systems perspective and not just from an individual component. According to the Software Coordinator, "the focus must be more on designing whole systems with reuse in mind." By finding generic designs across

several system applications, development personnel would be more apt to develop reusable designs and architectures.

b. Incorporating Ada Software Reuse in Project X

According to the literature review in chapter two, the Software Coordinator had to overcome five barriers to successfully implement the Ada reuse program. In summary, the five barriers are as follows: (1) reusable components must be of the highest quality to be economically beneficial, (2) reusable components cost substantially more to develop than customized components, (3) there must be sufficient top-down management support for the reuse program, (4) there must be a solid plan, procedures, tools, reusable materials, and methods in place, and (5) some type of measurement system must be used. The literature states that all of these barriers must be addressed and overcome before a software reuse plan can be properly incorporated [Ref. 7]. Alpha Organization management did succeed in addressing and overcoming the first two barriers, but failed in overcoming the third and fifth barriers. They were partially able to overcome the fourth barrier. In the next few paragraphs, we will analyze the organization's success or lack of success in overcoming these barriers.

Alpha Organization was easily able to overcome the first two barriers. High quality was one of the cornerstones of the current reuse program. Management and development personnel were completely committed to developing high quality components. The Software Coordinator new that higher costs were associated with the development of reusable components, but she also understood that higher initial development costs could result in cost savings through component reuse. Cost savings was a big driver in the Software Coordinator's decision to incorporate software reuse into Project X. According to the Software Coordinator, "using reuse to the maximum extent possible is the only way the organization will survive reduced

budgets and an increased emphasis on reduced costs." Literature information supports this premise that software reuse has the potential of reducing overall development costs [Ref. 22]. Reuse, when implemented as a formal systematic program, has the potential of increasing productivity, increasing quality, and reducing development times. All three of these potential benefits equates to reduced overall development costs. The impetus of the last two statements is that software reuse has the potential of reducing overall development costs. Designing reusable components does require more time because of increased emphasis on quality and coordination between software developers. The Increased development times result in increased development costs. However, remember that components designed for reuse can be used in other areas, thus saving overall development costs. The Software Coordinator and the Project Manager new the investment costs of developing reusable components would be higher, but they planned to reuse the components in other areas within Project X and future systems - thereby reducing long-term costs.

The organization was only able to partially overcome the fourth barrier. In her decision to incorporate software reuse into Project X, the Software Coordinator increased reuse support to development personnel. She anticipated that the increased support would reduce the organization's resistance to change. Software development personnel in the organization were comfortable with reuse, but only FORTRAN reuse. Personnel initially did not accept the Ada language. Furthermore, reuse support for Ada, in the form of a reuse library, was completely lacking. Most personnel did not want to invest the extra time and effort associated with learning the Ada language and developing Ada reusable components. To overcome this resistance, the Software Coordinator created an Ada Specialist position. The Ada Specialist, among other duties, would provide support to the organization's Software Reuse Coordinator and Reuse Committee on all matters relating to Ada reuse. Both

the Ada Specialist and the Software Reuse Coordinator provide support to Project X development personnel by searching other DoD Ada reuse libraries for required Ada reusable components. In addition to increasing reuse library support, the Software Coordinator also relaxed the strict requirements associated with the incorporation of developed Ada reusable components into the Ada reuse library. She anticipated that with management support, support from the Reuse Coordinator and Ada Specialist, and more relaxed reuse library process requirements, development personnel would be more willing to develop and use reusable components in Project X.

Despite good intentions, not all aspects of the fourth barrier were overcome. The literature stated that reuse must be planned for in advance by ensuring adequate materials and procedures are in place. These materials and procedures include: a reuse library, company standards and procedures, and design methods. As previously stated, software reuse procedures were in place along with supporting personnel, but development personnel many times circumvented these procedures because of project schedule pressures. These procedures, such as the incorporation of components into the Ada library, were time consuming and tedious. Programmers circumvented these procedures by informally acquiring reusable components from fellow programmers. Very few components were being introduced into the Ada library. Management was unaware of the extent of this problem. The result was a set of procedures that were only tokenly followed by programmers, and a less than adequate Ada reuse library. The availability of a more extensive Ada reuse library and stricter controls on procedures could have overcome the fourth barrier.

Very little was addressed by the organization's management in regards to overcoming the third and fifth barriers. Management support was lacking in many ways. The deliberate reuse program was highly supported by the organization's management, but schedule pressures were viewed by management as more important

than accepting increased development time associated with developing reusable components. Management's emphasis on schedule was the main reason development personnel circumvented the procedure of placing components into the Ada reuse library. More time should have been allocated on the schedule for the development and incorporation of reusable components into the Ada reuse library. Had this happened, development personnel would have better support for the reuse procedures.

Also lacking in the third barrier was management's commitment to educating software developers in planned software reuse. Both Programmer One and Programmer Two stated that development personnel do not know how to design reusable components that can be used in Project X and in future projects; at least not from the standpoint of deliberate and planned design for reuse. Programmers continue to use opportunistic/ad hoc reuse. This type of reuse is having a positive effect on Project X, but it is not helping to build a solid foundation of reusable components for future projects. In order to better secure the future of Ada reuse in the organization and in the maintenance of Project X, management must be more proactive in providing specialized training to development personnel on the deliberate and planned design of reusable software materials.

c. The Effect of Software Reuse on Project X

Software Reuse has generated both direct and indirect effects on Project X and Alpha Organization. Direct effects in Project X include: increased productivity, increased quality, and a greater percentage of reusable components. All the interviewees in this case study agree that the use of software reuse in Project X has increased productivity. The exact increase in productivity is unknown because Alpha Organization does not have a standing productivity measurement program in place. However, both Programmer One and Programmer Two said that their individual development productivity increased as a result of software reuse. Both

programmers also stated that the requirement for software reuse has forced them to develop higher quality components for Project X. However, once again, the exact increase in software quality is unknown because Alpha Organization does not effectively measure software quality.

Project X will be comprised of a large percentage of reusable components. The project is not yet completed, but the Project Manager estimates that 80% of the system will be comprised of reusable components. Programmer Two, who is presently working on the project, provides a more conservative estimate. She states that 50% or more of the system will be comprised of reusable components. Whether the amount is 80% or 50%, the fact remains that the requirement to use software reuse has helped productivity in Project X.

The requirement for software reuse has also produced an indirect effect on Project X and Alpha Organization. This indirect effect is associated with the development personnel's unwillingness to develop components to be processed into the reuse library. Software developers have outwardly supported the organization's formal deliberate reuse program, but they are still developing components using ad hoc algorithm and code sharing. The Ada reuse library has only moderately increased in size, due to a lack of support from software developers. This lack of support, as stated before, could be caused by development schedule pressures or a lack of training on planned software reuse. Whatever the cause, the fact remains that active support for the Ada reuse program must increase if Ada project development is to remain viable and productive in the future.

To summarize this section, the analysis on software reuse in Project X indicates that Alpha Organization has initiated a planned reuse program, but the plan is only tokenly supported by the software development personnel. Personnel are only developing reusable components for Project X with little regard for the components

applicability in future systems. Development personnel are refusing to design components that can be processed into the organization's Ada reuse library. This refusal could be due to either schedule pressures or a lack of reuse training. Also, many of these problems could be due to management's failure to address and overcome three of the barriers to implementing a viable formal reuse program. Whether it is through ad hoc practices or formal deliberate reuse, Project X has witnessed increased productivity and quality. The next section will address the opinions of the researcher in reference to Project X's three development requirements.

B. INTERPRETATION OF THE RESULTS

In this section, the researcher provides an interpretation of the results and analysis sections of the case study. Information presented reflects the opinion of the researcher. Similar to the last section, the opinions of the research is divided into three areas: (1) the Delta CASE tool on Project X, (2) the Ada programming language on Project X, and (3) software reuse on Project X. Information in this section is used in Chapter VI to answer the primary and subsidiary research questions.

1. The Delta CASE Tool on Project X

Upper level management chose not to include any project managers or software development personnel in the CASE tool selection process. In the opinion of the researcher, this decision did not coincide with a vision of success and it also failed to create buy-in among the Project Manager and software developers. The decision to procure the Delta CASE tool was made by two upper level managers within Alpha Organization without any input from the personnel who would be using the tool. The personnel ultimately responsible for using the tool at least should have been given the opportunity to test the Delta CASE tool before the contract was initiated. By allowing user testing, management would have been able to see first hand whether the tool was adequate for software development in the organization.

Also, considering the fact that the culture with Alpha Organization is so resistant to change, the organization would have achieved better buy-in from the software developers had these personnel been allowed to be a part of the selection process. According to the literature, the culture is a deciding factor as to whether CASE tools are utilized or should be utilized in an organization [Ref. 7].

Introducing the tool and the software engineering methods together was in itself risky. The Software Coordinator overestimated software developer's willingness to accept the complexity associated with the simultaneous introduction of a complex tool and software engineering methods. Additionally, prior to actually procuring the Delta CASE tool, the Software Coordinator made a series of decisions that further compounded the risk associated with the tool. These additional risks included the tool's inability to support Ada and its requirement for the unfamiliar OS/2 operating system. Programmer support for the Delta CASE tool ultimately suffered because of these compounding risks. Had the Software Coordinator involved development personnel in the CASE tool selection process from the beginning, the risk of acceptance by development personnel would have decreased. Development personnel should have been involved in the decision from the first time CASE tools were even considered for Alpha Organization. Additionally, the Software Coordinator should have procured a more simplistic tool; one that was easy to operate now, but could be updated with greater capabilities in the future. Management should have used the expertise of the developers to select the right tool and develop the required support for the tool.

Finally, in the opinion of the researcher, the Delta CASE tool was kept in the organization longer than it should have because of political reasons. The tool was a failure from the very beginning. All three pilot projects using the tool, were canceled due to reduced programmer productivity. Programmers assigned to these pilot

projects blamed the tool for the reduction in productivity. Programmer One said the tool was impossible to use. At this point, upper level management should have terminated the use of the Delta CASE tool in the organization. Instead, however, upper level management chose to use the tool in Project X even though the tool did not support the Ada language. Granted, upper level management's desire to get a return on the tool's large investment could have been the reason for the tool's continued use. However, upper level management also granted a waiver to the Project X Project Manager from using the tool. Subtle hints given to the researcher from certain interviewees indicated that the tool was used in Project X because certain proponents for the tool did not want to lose "public commitment to a course of action."

2. The Ada Programming Language on Project X

Although the Ada language is considered a success in Project X, it is the opinion of the researcher that Alpha Organization's management did not adequately prepare for the transition to Ada. When Alpha Organization management first began to implement Ada in the early 90s, the Ada language had already been mandated by DoD for over 10 years. Why did management not prepare more for the transition to Ada well in advance of Project X? Personnel should have been trained in the Ada programming language well before the start of Project X. An Ada reuse library should have been developed well in advance of the project. Software engineering principles, of which Ada so strongly supports, should have been incorporated into the organization's development process well in advance of the introduction of Ada. It is apparent there was little in the way of strategic planning within the organization in regards to preparing for the implementation of Ada. Had there been more prior planning and preparation for Ada, it is almost certain that the personnel and support problems of Project X would not have been so severe.

It is uncertain why Alpha Organization still used multiple languages in the design of Project X's parent Information Support System. In the opinion of the researcher, the Information Support System should have been developed with as few languages as possible. Operating and development budgets were getting smaller at this time. Upper level management, after the introduction of Ada, should have consolidated all software development under the Ada language and maybe one more other common standardized language. As previously stated, using a standardized language such as Ada means fewer interface problems for complex systems, lower development and maintenance costs through higher productivity and quality, and lower support costs. Ada continues to generate more support in DoD and in industry. With this increase in support, management should have pushed the implementation of Ada farther into the development of the parent Information Support System.

3. Ada Software Reuse on Project X

Software reuse, as a means to increase development productivity, is relatively successful for Project X. However, in the opinion of the researcher, there are several problems with Alpha Organization's Ada software reuse program. These problems are related to how Ada software reuse is being implemented by management and how it is being supported by development personnel. First and foremost, Alpha Organization's management was more reactive rather than proactive in developing the Ada reuse program. The program should have been created well before Ada was introduced into a critical system like Project X. Alpha Organization lacked adequate infrastructure and supporting mechanisms required to build a viable Ada reuse program. Supporting infrastructure like the Ada reuse library was also inadequate. Granted, the Software Reuse Coordinator and Ada Specialist are trying to increase the size of the library, but a large portion of this should have been completed before the start of Project X.

Alpha organization management also failed to provide most development personnel with the necessary skills and tools to support the deliberate design and development of reusable Ada components within Project X. Development personnel lack the necessary training in the Ada language. This is evident from the serious shortage of qualified Ada programmers in Project X. Personnel also lack development skills. According to Programmer One, most development personnel are unable to deliberately design Project X components under the auspice of software reuse. These development personnel also do not know how to build reusable components for the Ada reuse library. Only one or two experienced Ada programmers within Project X have developed components that can be incorporated into the Ada reuse library. The majority of reusable components in the project have been completed through ad hoc, opportunistic module sharing. Most of these ad hoc designed reuse modules have not been incorporated into the Ada reuse library.

The lack of modules being incorporated into the Ada reuse library brings up the second problem with the Ada software reuse program; development personnel are not supporting the reuse program. Personnel tell you they support the organization's reuse program and the concept of software reuse, but most of this support is "lip service." Personnel are not executing the deliberate design of reusable software as Alpha Organization management had intended. It is important to note that although development personnel are not adequately supporting the reuse program, management is also not providing adequate support, motivation, and incentives to development personnel. Schedule pressures are preventing development personnel from developing components that can be incorporated into the Ada reuse library. To change this trend and increase support of development personnel, management must allow for greater flexibility in development schedules. Additionally, management should develop an incentive program to entice Ada programmers to design modules

that can be incorporated into the library. An example incentive program would be to give a monthly monetary or recognition award to the software developer who successfully incorporated the greatest number of modules into the Ada reuse library. Furthermore, management should design a measurement program to measure the progress of Ada reusable components designed for Project X and the Ada reuse library.

C. SUMMARY

This chapter provided an analysis and interpretation of the interview results of the case study. Section A, the analysis portion of this chapter, compared the interview results on the three specific development requirements with the literature information in Chapter II. The procurement of the Delta CASE tool, the first of three development requirements, was a poor management decision that ultimately resulted in the tools failure. Upper level management chose to purchase a complex CASE tool that could not generate Ada code, had no buy-in from development personnel, and required the unfamiliar OS/2 operating system. The second requirement analyzed was the use of the Ada language in Project X. Ada was ultimately successful for Project X, but the implementation plan for introducing Ada into Alpha Organization was plagued with a shortage of qualified Ada programmers, low productivity, and higher operating costs. After Ada programmers became available and gained valuable experience, then productivity and software quality increased, and development costs decreased. The third requirement analyzed software reuse in Project X. The analysis indicates that Alpha Organization had initiated a planned reuse program, but the plan was only tokenly supported by software development personnel. The shortage of development personnel support was due to a lack of incentives and support from the organization's upper level management.

The second section in this chapter provided an interpretation of the results in chapter four and the information within the analysis section of this chapter. In the opinion of the researcher, Alpha Organization management made a grave error by not involving the Project X Project Manager and development personnel in the CASE tool selection process. Involving these personnel in the selection process would have created greater cultural buy-in and a better chance of procuring the right tool. In reference to the introduction of Ada into the organization and Project X, the researcher believes that management did not adequately prepare in advance for the transition to Ada. Support for Ada initially was lacking with both development personnel and management. Management should have implemented Ada programming training, software engineering principles, and a better Ada reuse library prior to introducing Ada into Project X. Had this been done, productivity may not have suffered initially within the project. Finally, the researcher interpreted the results and analysis information relating to software reuse in Project X. Software reuse was considered relatively successful for Project X, but there were problems with management's software reuse implementation plan and with development personnel support for the plan. The Ada software reuse implementation plan, similar to the Ada language implementation plan, did not adequately prepare development personnel for the tasks of deliberately designing Project X for reuse. Personnel lacked training in the Ada language and in the deliberate design of reusable components. Very few components were incorporated into the Ada reuse library. Most components were constructed as a result of ad hoc opportunistic module sharing between programmers. Management could have prevented this problem by developing an Ada software reuse incentive program, and through adequate motivation and support for development personnel. In Chapter VI, the information presented in this chapter and the last

chapter is used to answer the primary and subsidiary research question. Concluding comments is also provided in Chapter VI.

VI. CONCLUSION

This thesis study researched three specific development requirements of Project X, and the effect of these requirements on the development of the project. A case study research methodology was employed in this thesis study. Information obtained from interviews with key personnel involved in Project X was analyzed and interpreted within the scope of the thesis research questions. In the first section of this chapter, the primary and subsidiary research questions are summarily answered. The second section provides recommendations for further research. Finally, the last section of this chapter provides concluding statements to the thesis study.

A. ANSWERS TO RESEARCH QUESTIONS

This section provides summarized answers to the primary and subsidiary research questions presented in Chapter I of this thesis. The primary research question was addressed in this thesis study through exploration into the four subsidiary research questions. As such, answers to the subsidiary research questions are provided first, followed thereafter by the primary research question.

1. Subsidiary Research Questions

The first subsidiary research question for this thesis is: ***Why was the Delta CASE tool used and what was its impact on Project X in terms of productivity?***

- Alpha Organization upper level management wanted a tool that could automate all phases of the software development process for Project X and other development projects in the organization.

- The goal was to simultaneously introduce the Delta CASE tool with a specified software engineering methodology in order to standardize the development process. Another goal of management was to achieve higher development productivity with the tool by creating a development process that is less variable and more predictable.

- Management saw the Delta CASE tool as a way to make development personnel follow standards.

- The Delta CASE tool negatively impacted Project X. This negative impact was directly caused by the following problems:

  * The Software Coordinator went against the recommendations of industry by introducing the methods and the Delta CASE tool at the same time. Development personnel said the tool was too difficult to use with the specified software engineering methods. Personnel were more apt to blame the tool instead of the methods for their development difficulties.

  * Development personnel were required to learn and use the unfamiliar OS/2 operating system. The requirement for OS/2 greatly increased Delta CASE tool investment costs.

  * The Delta CASE tool could not be used to create Ada code. One of the initial acquisition requirements stated that the tool be able to generate code.

  * Development personnel did not receive adequate formal training and support on the Delta CASE tool. These personnel became more dissatisfied with the tool.

- Productivity in Project X decreased because of the Delta CASE tool.

- The Project X Project Manager requested and was granted a waiver from using the Delta CASE tool because of the decreased productivity.

The second subsidiary research question for this thesis is: ***Why was the Ada programming language chosen and what was its effect on Project X in terms of productivity, quality, and cost?***

- The use of the Ada programming language in Project X was mandated by DoD.

- Even without the mandate, the Software Coordinator still would have chosen Ada for the following reasons:

  * It is an internationally recognized standard programming language.

  * Ada encourages development personnel to follow software engineering methods.

  * Software written in Ada is easier to maintain.

  * Ada has higher investment costs, but these costs are paid for through lower life cycle costs. As Ada increases market share in industry, investment costs will become lower.

  * The structure of Ada makes the language well suited for developing reusable software components.

- Alpha Organization upper level management developed an implementation plan to slowly introduce Ada into the organization's development process. Project X was the first major project in Alpha Organization using Ada.

- The introduction of Ada into Project X initially reduced the project's development productivity for the following reasons:

  * Development personnel were not trained in the Ada language.

  * Alpha Organization had difficulty in contracting out for qualified Ada development personnel. Contract development personnel did not have training in both the Ada language and software engineering principles. The Project Manager was forced to invest extra time and resources to train personnel before they could be adequately used in the project.

  * Development personnel had no experience in the deliberate design of Ada reusable software. A lack of experience with the Ada language itself was partially to blame. Also to blame was the inadequacy of the Ada reuse library.

- Development personnel stated that Ada components in Project X have higher quality than other components developed in the past.

- Development costs for Ada components developed in Project X are higher than components developed in past projects. Increased development costs are due to increased development time and increased training and supporting equipment costs.

- Life cycle costs for Project X Ada components are lower than components in other projects because Ada components are easier to maintain, and they are designed to be reused in other applications.

The third subsidiary research question for this thesis is: ***What was the extent of software reuse in Project X and what was its effect on the project in terms of productivity and quality?***

- Alpha Organization designed a formal deliberate software reuse program for the Ada programming language. The program is designed to support the development of reusable components for Project X and other projects using Ada.

- The organization established an Ada software reuse library as supporting infrastructure for the software reuse program. A reuse committee chaired by a Software Reuse Coordinator and staffed, among others, by an Ada Specialist, is responsible for ensuring the usability of the Ada reuse library for project development personnel.

- The reuse committee is responsible to screen newly developed Ada components for possible induction into the Ada reuse library.

- Project X development personnel verbally supported the program, but most of these developers are not willing to develop components for induction into the reuse library. Personnel state that the component induction process is too time consuming and requires strict adherence to quality and documentation requirements.

- Most software reuse in Project X consists of black-box source code reuse. Development personnel create reusable components through informal ad hoc module sharing.

- Between 50% and 80% of Project X will consist of reusable components.

- Software reuse had a positive effect on productivity in Project X. Components take longer to develop, but they are able to be used again in several different areas, resulting in an overall increase in development productivity.

- However, there were several problems with the reuse program in relation to productivity. Productivity would have been higher in Project X and future Ada projects had the following problems been corrected:

  * The creation of a larger more usable Ada reuse library prior to the start of Project X.

  * Provide development personnel with training in how to deliberately design and develop reusable components for induction into the reuse library.

  * Alpha Organization upper level management should have provided development personnel incentives to induct components into the library.

  * Management desperately needs a measurement program to measure development productivity associated with software reuse.

- As previously stated, reusable components designed for Project X are higher quality than custom designed components.

The fourth subsidiary research question for this thesis is: ***What corrective actions, in reference to the three subject development practices, could have been implemented to improve the development process?***

- It is obvious that the Delta CASE tool was inadequate for Project X. The Delta CASE tool was too difficult to understand, learn, manipulate, and use. Management did not adequately identify language requirements for future projects during the time CASE tools were being considered for Alpha Organization. Management also underestimated development personnel's willingness to accept software engineering methods and a new complex CASE tool.

- The following corrective actions concerning the Delta CASE tool could have been implemented by Alpha Organization upper level management to improve the development of Project X:

  * The Project Manager and representative development personnel should have been involved early and completely in the CASE tool selection, testing, and fielding processes.

  * Management should have started with a simpler CASE tool; a CASE tool which could be upgraded with more complex features as development personnel gained operating skills and confidence.

  * If available, management should have procured a CASE tool that supports the Ada language. At the very least, management should not have procured a tool that does not support Ada.

  * Management should have chosen a CASE tool that uses the same operating system required by other systems in the organization.

  * Management should have incorporated software engineering methods into the organization's development process well before the introduction of the CASE tool.

  * Development personnel should have been given time at their own pace to become confident with the CASE tool's operation before it was introduced into Project X.

- The incorporation of Ada into Project X was plagued with problems. Even though Ada's use in Project X was mandated by DoD, management still had ample time to prepare for its arrival into the organization. Management's actions towards Ada were more reactive rather than proactive.

- The following corrective actions concerning the introduction of Ada into Alpha Organization could have been implemented by upper level management to improve the development process in Project X:

  * Management should have trained development personnel in Ada well before the language was introduced into Project X. There would not have been any Ada programmer contracting requirements had management trained organizational development personnel in Ada.

  * Management should have stabilized both the software engineering methodology and the Ada language in the organization's development process prior to initiating Project X.

  * A more extensive Ada reuse library should have been available to development personnel prior to the start of Project X.

  * Management should have procured a CASE tool that supports the Ada language.

- Alpha Organization had a significant formal reuse program in place, but a majority of the reuse infrastructure in the program only supported the FORTRAN language.

- The Ada reuse supporting infrastructure in the organization was minimal at the start of Project X. Furthermore, most development personnel were not adequately executing the requirement to design and develop Project X under the auspice of software reuse.

- Alpha Organization upper level management could have implemented the following corrective actions to make software reuse more beneficial within the development of Project X:

  * Development personnel should have been formally trained in the deliberate design of systems (and the systems' components) for reuse.

  * Management should have developed an incentive program to induce Project X developers into formally developing and incorporating reusable components into the Ada reuse library.

  * Management should have allotted more time in the development schedule to provide development personnel ample time to formally design and develop reusable components for induction into the Ada reuse library.

  * The process of placing reusable modules into the reuse library should not have been so confrontational. Management should have established a teaming arrangement to help Project X development personnel develop and incorporate reusable modules for the Ada reuse library.

- Upper level management should have teamed up with the Project Manager and development personnel earlier in the development stages of the implementation plan. Establishing a management/developer teaming approach would have created more personal buy-in with the new requirements and less resistance to required changes in the development process.

2. Primary Research Question

The primary research question for this thesis is: ***How did management's decision to implement -- the Delta CASE tool, the Ada programming language, and software reuse -- affect the development of Project X?***

- Upper level management's decision to introduce these three development requirements into Project X was sound. However, there were notable problems with management's implementation plan.

- Several factors of the implementation plan initially resulted in lower than expected development productivity for Project X. Additionally, several of these factors significantly increased the investment costs associated with project development. The factors and their associated effects are as follows:

  * Management introduced a CASE tool (the Delta CASE tool) before Project X's language requirements had been firmly established. Development personnel could not use the tool for all phases of the development process because the tool did not support Ada.

  * Management made a risky decision of introducing the Delta CASE tool and software engineering methods at the same time. This risky venture required personnel to completely change from an ad hoc to a more structured development process. Personnel struggled with the tool and the new methodology. Most programmers blamed the tool for a reduction in development productivity.

  * The Delta CASE tool's requirement for the OS/2 operating system resulted in higher investment costs and greater change accommodation from development personnel.

  * Management did not adequately prepare for the introduction of Ada. Personnel did not receive Ada training well in advance of Project X. There were serious shortages of qualified Ada programmers in the organization. Even contract support Ada programmers lacked necessary skills to continue development. Productivity initially suffered until personnel became more familiar with Ada.

  * Development personnel did not receive formal training in the deliberate design of systems and components for reuse. Most

development personnel developed components through ad hoc development procedures.

* The organization did not have an adequate Ada reuse library when Project X development was initiated. Personnel refused to incorporate components into the Ada reuse library because the process was too time consuming and requirements were too strict.

- The bottom line is that Alpha Organization upper level management attempted to do too much in their implementation of the three development requirements for Project X. Development personnel were required to accommodate too many changes too soon before the start of Project X.

- Despite the problems associated with several factors of the implementation plan, the project is now basically on schedule. The develop schedule was the only project output that management could effectively measure. The Project Manager and development personnel were able to overcome problem areas such as increased development costs and an initial shortage of qualified Ada programmers. Although subjectively measured, productivity did increase once programmers became familiar with Ada. These programmers were effectively able to bring the project basically back on track following the increase in productivity.

B. RECOMMENDATIONS FOR FURTHER RESEARCH

The following areas are recommended for further research:

1. **Investigate the current availability of Ada CASE tools and the market trends of Ada CASE tool development within the commercial sector.**

As Ada continues to be used in the development of DoD and commercial applications, so to will the demand for quality CASE tool support. Additional research should look at the availability of Ada CASE tool vendors and the quality of

their products. Research should also look at recent trends in industry in regards to available Ada CASE tool vendor support. The Integrated CASE tool established for DoD shows outstanding potential for Ada development support. Further research could investigate the ICASE for its applicability in both DoD and commercial development processes.

2. **Investigate the current and future availability of Ada support in the commercial sector, industry, and in educational institutions.**

The fate of the Ada language in future software development is still questionable. Without necessary support from -- industry, commercial vendors, and educational institutions,-- Ada can not have a promising future in software development. Additional research in these areas should be focused on the following: what is the current and future availability of vendors offering Ada support products, what is the current and future availability of personnel trained and experienced in Ada (user base), and what is the current and future availability of educational institutions offering curricula in the Ada programming language.

3. **Investigate the current and future use of Ada 95 in both DoD and commercial software development projects.**

Ada 83, the first standardized version of Ada, has interfacing limitations and does not lend itself to object oriented design. Ada 95 is a big improvement on Ada 83. In addition to other state of the art features, the new Ada language has corrected the language interface problem and it supports object oriented design. The vast potential for Ada 95 in future software development identifies the following areas of further study: what are the inherent advantages of using Ada 95, what current and future development projects within DoD and industry will use Ada 95, how does Ada 95 compare to Ada 83 or other languages presently being used in DoD and the commercial sector.

4. **Investigate the extent of DoD's centralized Ada software reuse repository.**

As defense budgets get smaller, more emphasis is being placed on reducing software development costs. Software reuse, when implemented as a formal deliberate program, is a cost effective way to reduce development costs through increased productivity and increased quality. Through the use of DoD's centralized Ada software reuse repository, software development organizations have the means to retrieve necessary components for current and future software development projects. Potential areas of further study on DoD's central repository should include the following: what are the procedures for requesting module support from DoD's centralized libraries, what are the procedures for incorporating developed modules into the libraries, identify to what extent DoD organizations use the libraries for support, and what is the future plan for DoD centralized Ada reuse support.

C. CONCLUDING REMARKS

This thesis addressed the impact of three specific development requirements on a representative DoD software development project. Software continues to be one of the greatest areas of risk associated with system development. As demand for software intensive systems grows, so too will the risk associated with software development. Finding the means to control risk will become even more difficult as defense budgets continue to decrease in size. Management must find ways to identify and control potential risk in order to mitigate the effects of cost, schedule, and performance on software development projects.

The case analysis of Project X identified specific risks and problems associated with the project's development requirements of using a particular CASE tool, the Ada language, and software reuse. CASE tools are designed to enhance productivity in the development process. Since the design characteristics of the Delta CASE tool did

not match the development requirements of Project X, the tool had the opposite effect of reducing development productivity in the project. Ada as a development language promises higher productivity, higher quality, and greater potential for reuse. However, as a development requirement for Project X, Ada had the initial effect of reducing productivity until development personnel became skilled with the language. The requirement of software reuse is the wave of the future promising lower costs, higher productivity, and higher quality. Software reuse in the development of Project X was beneficial, but it never reached its full benefit potential. With the exception of the Delta CASE tool, the problems with this case study were not in the actual development requirements, but in how the requirements were implemented into the development process. As is evident from the results of this case study, the importance of thorough prior implementation planning in reducing risks and potential problem areas cannot be overemphasized. Upper level management must remain totally focused and committed to implementation planning, one of the most important steps of bringing about successful organizational change.

Through case study analysis of software development projects, researchers can explore the effects of specific development processes/requirements and acquisition strategies on producing quality systems on time and within budget. Lessons learned from these case study analyses can be used to correct deficiencies in other development processes. As the need for quality, reliable, and affordable software continues to rise, so to will the importance of refining software development processes for future system development.

APPENDIX. INTERVIEW QUESTIONS

Questions given to all interviewees

1. What is your job title and job description?

2. How long have you been with Alpha Organization?

3. How long have you been in the software development domain?

4. What is your previous experience in software development?

5. What is your formal education background?

6. Do you have any formalized training in software engineering?

7. Do you have any formalized training in software development management?

8. What type of experience do you have with CASE tools? What classes of CASE tools did you use (i.e., editing, programming, verification and validation, configuration management, metrics and measurement, project management, and miscellaneous tools)?

9. Do you have any formalized training on CASE tools?

10. Does the use of CASE tools in this project and others improve the development process through the application of software engineering principles and greater automation?

11. Do CASE tools improve productivity? Are less people required as a result?

12. Does the culture within Alpha Organization embrace the use of CASE tools?

13. Does the mission of your organization have any bearing on CASE tool use or lack of CASE tool use?

14. Did the size of the project or the complexity of the project have any bearing on the decision to use or not use CASE tools?

15. Do you personally feel CASE tools improved the development of Project X?

16. What type of language had you used in the past and how does Ada compare?

17. Is there a language that would have been more suited to this project? In what ways?

18. What is your background in Ada?

19. Did you have any problems finding qualified programmers? Is there an industry wide shortage of experienced Ada programmers?

20. Are there higher investment costs associated with using Ada?

21. What are some of the benefits of using the Ada programming language in Project X?

22. How did the use of Ada improve productivity, quality, performance, and software reuse?

23. Do development personnel support the idea of using Ada?

24. Should Ada be considered for use in future projects within Alpha Organization?

25. What level of experience do you have in software reuse?

26. Does Alpha Organization have a formal deliberate reuse program established?

27. What is the extent of Alpha Organization's reuse program?

28. What facet of reuse is utilized within Alpha Organization?

29. Has a reuse library been developed?

30. What percentage of Project X will be comprised of reusable components?

31. Do you think software reuse is cost effective for Alpha Organization?

32. Are enough reusable materials developed to warrant deliberate design for reuse?

33. Does reuse increase development productivity and quality of software?

34. Does reuse reduce maintenance requirements?

35. Do programmers embrace the use of software reuse?

36. Does Alpha Organization's development methodology support software reuse?

37. Is the performance of Project X affected by the three development requirements?

Questions given only to the Programmers

38. In what other ways did CASE tools improve the development process?

39. Does management support the idea of using Ada?

40. What level of support does management provide for CASE tool use?

41. What level of support does management give to software reuse, and the development of a deliberate software reuse program?

42. Do the three development requirements complement each other in their ability to create quality software?

Questions given only to the Software Coordinator and the Project Manager

43. Were there any contractual obligations for using a particular CASE tool?

44. Did your organization choose Ada because it was mandated or because it was a better language?

45. What were some of the reasons your organization chose Ada?

46. Are there higher investment costs associated with using Ada?

47. Did lower costs, higher quality, lower maintenance costs, and language integration capabilities drive the decision to use Ada?

48. What steps were taken to develop and implement the reuse program?

49. Is Project X under or over budget, and did the three development requirements have any effect on this outcome?

50. Is Project X on or off schedule, and did the three development requirements have any effect on the schedule?

51. Did the selection of Ada as the programming language for Project X have any bearing on the decision to use a specified CASE tool and software reuse?

LIST OF REFERENCES

1. Kitfield, James, "Is Software DoD's Achilles' Heel?", *Military Forum,* July 1989.

2. McCaffrey, M. J., Visiting Assistant Professor, Department of Systems Management, Naval Postgraduate School, *Management of Embedded Weapon System Software,* Graduate Course for Systems Acquisition Curriculum, Spring Quarter 1995.

3. Mosemann, L. K. II, Deputy Assistant Secretary of the Air Force, *Guidelines for Successful Acquisition and Management of Software Intensive Systems,* Software Technology Support Center, Department of the Air Force, Vol. 1, September 1994.

4. Tate, G., Verner, J., Jeffery, R., "CASE: A Testbed for Modeling, Measurement and Management", *Communications of the ACM*, Vol. 35, No. 4, April 1992.

5. Fuggetta, A., Politecnico di Milano and CEFRIEL, "A Classification of CASE Technology", *IEEE Software*, December 1993.

6. Bauer, C. J., "CASE Tools", *Government Computer News*, Vol. 14, No. 7, April 1995.

7. Jones, Capers, *Assessment and Control of Software Risks*, PTR Prentice Hall, 1994.

8. Graham, Carol, "CASE Cracks Applications Backlog", *DATAMATION*, Nov., 1994.

9. Rai, A., Howard, G. S., "Propagating CASE Usage for Software Development: An Empirical Investigation of Key Organizational Correlates", *OMEGA*, Vol. 22, No. 2, 1994.

10. Sprague, R. H. Jr., McNurlin, B. C., *Information Systems Management In Practice,* Prentice-Hall, Inc., 1993.

11. Constance, P., "Survey Confirms Ada is top DoD language; mandate prevails, and Cobol is fading", *Government Computer News*, Vol. 14, No. 9, May 1995.

12. United States General Accounting Office, Report to the Chairman, Subcommittee on Defense, Committee on Appropriations, House of Representatives, *Programming Language, Defense Policies and Plans for Implementing Ada*, U. S. Government Printing Office, September 1991.

13. Hinden, H. J., Rausch-Hinden, W. B., "Real-Time Systems", *Electronic Design*, 1983.

14. Anderson, C., "Ada 9X Project Management", *Communications of the ACM*, Vol. 35, No. 11, November 1992.

15. McCarthy, S. P., "Ada 95 hits the street running, new updates to be incremental", *Government Computer News*, Vol. 14, No. 5, March 1995.

16. Wolfe, A., "Ada ain't dead", *Electronic Engineering Times*, No. 852, June 1995.

17. Ploedereder, E., "Building Consensus for Ada 9X", *Communications of the ACM*, Vol. 35, No. 11, November 1992.

18. Sikorovsky, E., Monroe, J. S., "DoD, DoE push commercial Ada", *Federal Computer Week*, August 1995.

19. Constance, P., "Teach Ada, win cash", *Government Computer News*, Vol. 14, No. 12, June 1995.

20. Prieto-Diaz, R., Reuse, Inc., "Status Report: Software Reusability", *IEEE Software*, May 1993.

21. Anthes, G. H., "Software reuse plans bring paychecks", *COMPUTERWORLD*, December 1993.

22. Frakes, W. B., Isoda, S., "Success Factors of Systematic Reuse", *IEEE Software*, September 1994.

23. Kaspersen, D., Unisys, "For Reuse, Process and Product Both Count", *IEEE Software*, September 1994.

24. Lim, W. C., "Effects of Reuse on Quality, Productivity, and Economics", *IEEE Software,* September 1994.

25. Schlukbier, A. C., "The future is reuse", *COMPUTERWORLD*, May 1995.

26. Joos, R., Motorola, "Software Reuse at Motorola", *IEEE Software*, September 1994.

27. Interviewee One, Organization Software Coordinator, Alpha Organization, interview with author, 26 January 1996.

28. Interviewee Two, Project Manager, Project X, Alpha Organization, interview with author, 25 January 1996.

29. Interviewee Three, Programmer One, Project X, Alpha Organization, interview with author, 25 January 1996.

30. Interviewee Four, Programmer Two, Project X, Alpha Organization, interview with author, 25 January 1996.

31. Project Management Office, Project Manager, Project X Specification Document, 1995.

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center 2
   8725 John J. Kingman Road, Suite 0944
   Fort Belvoir, VA 22060-6218

2. Dudley Knox Library 2
   Naval Postgraduate School
   411 Dyer Road
   Monterey, CA 93943-5101

3. Director, Training and Education 1
   MCCDC, Code C46
   1019 Elliot Rd.
   Quantico, VA 22134-5027

4. Defense Logistics Studies Information Exchange 1
   U.S. Army Logistics Management College
   Fort Lee, VA 23801-6043

5. Prof. David V. Lamm (Code SM/Lt) 5
   Naval Postgraduate School
   Monterey, CA 93943-5103

6. Prof. Nancy C. Roberts (Code SM/Rc) 2
   Naval Postgraduate School
   Monterey, CA 93943-5103

7. Prof. Tarek K. Abdel-Hamid (Code SM/Ah) 1
   Naval Postgraduate School
   Monterey, CA 93943-5103

8. Loren J. Dugan 2
   214 Kings Way
   Kalispell, MT 59901